THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Theoretical simulations of dynamical systems for advanced reservoir computing applications

VASILEIOS ATHANASIOU

Department of Microtechnology and Nanoscience - MC2
Chalmers University of Technology
Gothenburg, Sweden, 2020

Theoretical simulations of dynamical systems for advanced reservoir computing applications
VASILEIOS ATHANASIOU
ISBN 978-91-7905-293-5

Doktorandavhandlingar vid Chalmers tekniska högskola
Ny serie nr 4760
ISSN 0346-718X

Department of Microtechnology and Nanoscience – MC2
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone + 46 (0)31-772 1000

Cover:

Printed in Gothenburg, Sweden 2020

Theoretical simulations of dynamical systems for advanced reservoir computing applications

Vasileios Athanasiou

Department of Microtechnology and Nanoscience - MC2

Chalmers University of Technology

Abstract

There are computational problems that are simply too complex and cannot be handled by traditional CMOS technologies due to practical engineering limitations related to either fundamental physical behavior of devices at small scales, or various energy consumption issues. The field of unconventional computation has emerged as a response to these challenges. Up to date unconventional computation encompasses a plethora of computing frameworks, such as neuromorphic computing, molecular computing, reaction-diffusion computing, or quantum computing, and is ever increasing in its scope. This thesis is biased towards developing sensing applications in the unconventional computing context. This initiative is further extended towards developing novel machine learning applications.

The possibility of building intelligent dynamical systems that collect information and analyze it in real-time has been investigated theoretically. The basic idea is to expose a dynamical system to the environment one wishes to analyze over time. The system operates as an environment sensitive reservoir computer. Since the state of the reservoir depends on the environment, the information about the environment one wishes to retrieve gets encoded in the state of the system. The key idea exploited in the thesis is that if the state of the reservoir is highly correlated with the state of the environment then the information about the environment can be inferred with a modest engineering overhead.

A typical dynamical system is assumed to be a network of environment sensitive elements. Each element can be something simple, but taken together, the elements acquire collective intelligence that can be harvested. These ideas have been examined theoretically (and verified experimentally) by simulating various networks of environment-sensitive elements: the memristor, the capacitor, the constant phase element and the organic field effect transistor element. The simulations were done in the context of ion sensing, which is an extremely complex, many-body, and multi-scale modeling problem.

"It always seems impossible until it's done "

Nelson Mandela

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Zoran Konkoli for the continuous support of my doctoral studies and related research, for listening to me and spending time for discussions, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better supervisor for my doctoral studies.
I thank my co-supervisor Dag Winkler for his useful comments.
I thank the collaborators of the RECORD-IT project for the stimulating scientific discussions and for supplying me with useful experimental data.
Last but not the least, I would like to thank my family: my wife Christina for spending her life with me and supporting me, my son Alexandros who was born almost in the beginning of my doctoral studies and gave me the power to do research for a better future of humanity, to my parents and my brother's family for supporting me spiritually throughout writing this thesis and my life in general.

Contents

List of Abbreviations

AC	alternating current
CMOS	complementary metal-oxide-semiconductor
CPE	constant phase element
DC	direct current
ECG	electrocardiograms
HAT	hours between hospital and intensive care unit admittance
HR	heart rate
ICU	intensive care unit
ICULOS	length of stay in intensive care unit
MFB	memristor element connected with a delay feedback mechanism
MNA	modified nodal analysis
NaN	not a number, missing value of a data set
NOMFET	nanoparticle organic memory field-effect transistor
ODE	ordinary differential equation
OECT	organic electrochemical transistor
OT-sensor	oxytocin sensor
O2sat	blood oxygen level saturation
PoA	Point of Acquisition
PM	parallel connection of memristors
QRS	QRS peak in electrocardiograms
RC	resistor-capacitor
RECORD-IT	reservoir computing with real-time Data for future IT technologies
RLC	resistor-inductor-capacitor
RNN	recurrent neural network
SC	sensing capacity
SWEET	state weaving environment echo tracker
Temp	Temperature

List of publications

I. Vasileios Athanasiou & Zoran Konkoli (2017) On using reservoir computing for sensing applications: exploring environment-sensitive memristor networks, International Journal of Parallel, Emergent and Distributed Systems, DOI: 10.1080/17445760.2017.1287264

II. V. Athanasiou, Z. Konkoli, On the use of collaborative interactions for embedded sensing applications: Memristor networks as intelligent sensing substrates, submitted 2020, Under review

III. Athanasiou V, Konkoli Z. On the efficient simulation of electrical circuits with constant phase elements: The warburg element as a test case. Int J Circ Theor Appl. 2018;46:1072–1090. https://doi.org/10.1002/cta.2474

IV. Athanasiou V., Pecqueur S., Vuillaume D., Konkoli Z., On a generic theory of the organic electrochemical transistor dynamics, Organic Electronics, Volume 72, 2019, Pages 39-49, ISSN 1566-1199, https://doi.org/10.1016/j.orgel.2019.05.040

V. Athanasiou V., Konkoli Z. (2019). 'On mathematics of Universal Computation with Generic Dynamical Systems' , in Adamatzky A., Akl S., Sirakoulis G. Ch. (ed.) From Parallel to Emergent Computing. CRC Press Taylor and Francis Group

VI. Athanasiou V., Konkoli Z. (2019) Memristor Models for Early Detection of Sepsis in ICU Patients, Computing in Cardiology 2019, Volume 46, ISSN 2325-887X, 0.22489/CinC.2019.223

VII. Athanasiou V., Tadi K. K., Hurevich M., Yitzchaik S., Jesorka A., Konkoli Z. On sensing principles using temporally extended bar codes, IEEE sensor journal, https://doi.org/10.1109/JSEN.2020.2977462

VIII. Athanasiou V., Konkoli Z. On Improving The Computing Capacity of Dynamical Systems, submitted 2019, under review

Chapter 1

Introduction

Moore's law predicts that the number of transistors in a chip roughly doubles every second year. [1] However, it is likely that this trend will slow down due to practical engineering limitations or specific physical effects pertinent to small scales, such as wiring problems or electron tunneling. There are problems that are simply too complex and cannot be handled by traditional CMOS technologies. In somewhat technical terms, there are information processing applications that do not scale according to the Moore's law. Typical examples include problems in distributed, real-time, or embedded information processing applications. Accordingly, there is a need to develop alternative information processing solutions by using non-CMOS based technologies, with the additional benefit of low power computation, depending on the hardware used. In the information processing industry, and especially semiconductor industry, *functional diversification* has emerged as the field where there is a need to integrate non-CMOS devices with traditional CMOS devices.[2] The field of unconventional computation has emerged as a response to this functional diversification challenge. Up to date unconventional computation encompasses a plethora of computing frameworks, such as neuromorphic computing, molecular computing, reaction-diffusion computing, or quantum computing, and is ever increasing in its scope. [3, 4, 5, 6, 7, 8]

In particular, reservoir computing has gained a considerable interest among the unconventional computation community in the recent decade, both as a model of computation and as a remarkably practical approach for realizing neuromorphic computations. Historically, the field of reservoir computing started as an insight about behavior of synaptic weights during the neural network training process. [9, 10, 11, 12] In the field of supervised learning the weights need to be adjusted to achieve a desired computation. In reservoir computing, it has been realized that only a limited set of weights needs to be adjusted in the network training process. While it is true that a neural network is essentially a geometry free object, the weights that need to be adjusted belong mostly to the links in the network that can be naturally described as an "outer layer".

This led to the further insight that instead of a neural network one could use an arbitrary dynamical system as the core, and augment it with an outer layer. The dynamical system used this way is referred to as a reservoir, and the outer layer is referred to as the readout layer. The modern understanding of reservoir computing emphasizes the fact that a Turing universal expressive power can be achieved in the context of time-series data processing, if the reservoir exhibits separation property on the state of inputs. This separation property is rather generic, it is not limited in the engineering sense, and is often realized by complex systems at the edge of chaos. [13]

Though the foundational ideas behind reservoir computing have been around for quite some time, this is still an aggressively developing field which is gaining in

momentum. A reservoir computer is essentially a pattern recognition device. It consists of two parts, a dynamical system, referred to as the reservoir, that can be driven by external signals, and an easily trainable readout layer. The dynamical system has memory properties, *i.e.* the current state of the system depends on its own history. The external signals consist of the input accepted by the system. By assumption, the readout layer is the only part of the device that can be optimised. The external signals drive the system to a specific region of the configuration space, which constitutes the act of computation since the information stored in the external signals is transformed into the information stored in the internal state of the reservoir. The term "reservoir" stands for the reservoir of states. The readout layer has access only to instantaneous values of the state and extracts information related to the external signals.

The key claim is that any computation can be realized this way, provided the dynamical system is complex enough. Due to the inherent flexibility and the ease of use the reservoir computing approach is being applied frequently in many applications that require neuromorphic computation. [14, 15] The reservoir computer can be used with a minimum of preparation as an artificial intelligence unit that processes external information. From the theoretical point of view, the specific feature that makes reservoir computing popular is the ease of training. Likewise, from the engineering point of view, the readout layer can be a relatively simple structure.

The work done in this thesis has been strongly aligned with the activities in the RECORD-IT research project. In this project, it has been investigated whether it is possible to use reservoir computing to build intelligent sensing devices that can collect and analyze information simultaneously. The key idea is that the environment it is wished to sense interacts with a dynamical system, which is referred to as a *sensing reservoir*, or a *state weaver*. The sensing reservoir accumulates, or weaves in, information about the environment into its internal state over time. In such a setup the flow of information is not linear (from the sensor to the artificial intelligence unit), but the sensor and the associated information processing intelligence are the same.

The work of this thesis can be used to develop intelligent bio-compatible devices sensitive to changes of ion concentrations in their surrounding environments. Thus, many of the ideas presented in the thesis have been motivated by a need to model and simulate the network elements of interest for the RECORD-IT project, in the following to be referred to as *experimental elements*. Such elements can be described as wet nanoparticle organic memory field-effect transistors (NOMFETs), coated Si nanowires, self-conjugated polymers, or arrays of photocells. The state of those elements depends on ion concentrations of their surrounding environment.

Additionally, most of these elements have memory properties because their state depends on its own history. This property makes them useful building blocks for solving temporal information processing problems, such as time series classification or predictions. Due to this property, their state depends also on the history of their environment. In the context of this thesis, the environment is considered as ion concentrations in their surrounding aqueous solution. Therefore, information about the history of ion concentrations could be encoded in their state. Decoding this information could be done by analysing their state with a readout layer.

Elements exhibiting these properties can be combined to build powerful sensing devices. A natural theoretical paradigm for modelling these structures is a network of environment sensitive electrical components. This paradigm can also be used for temporal processing problems, such as time series classifications and predictions. These options define the context of reservoir computing which this thesis work has been done.

1.1 Aim and Scope

The overarching aim has been to study theoretically the possibility of using networks consisting of *experimental elements* for neuromorphic computations. This is done in two directions. In the first direction, networks are considered as environment sensitive electrical circuits to be used for extracting information regarding the environment temporal behavior. A key feature that is investigated is how coupling between elements affects the sensing capacity of a network. Additionally, it is studied whether sensing capacity improves by supplying an external drive signal. Similar ideas have been considered in the field of traditional machine learning (feature engineering). [16] To simulate accurately the transient dynamics of such networks, mathematical models of some elements have been developed because such accurate and efficient models have not been found in literature. In the second direction, networks of experimental elements are considered for temporal information processing problems such as the traditional time-series classification and prediction. Key findings from the first direction have been useful for developing methods in this second direction.

The key idea is that spatial-temporal information of the environment can be accumulated in the state of a reservoir if the system is arranged properly. [17, 18] This could happen if small pieces of information that are scattered over time (and that might be ignored in a standard sensing setup) can be accumulated, amplified, and ultimately stored in the state of a network. When should one use such a device?

The use of such devices would be advantageous in situations where information processing is necessary at the point of acquisition (PoA), *e.g.* at the side of sensors where analog signals are obtained. Information can be extracted at the PoA by *in situ* pre-processing, real-time analysis options can be provided, and accordingly a whole network of sensors can become more responsive. [19] Moreover, from the engineering point of view, such sensing solutions could be more flexible and easier to implement, reduce the necessary communication bandwidth, reduce energy consumption, and be bio-compatible.

In reservoir computing, internal weights of networks are not trained. Therefore, gradient descent optimisation techniques are avoided and so do problems of exploding and vanishing gradients. Additionally, training time can be relatively small since a reservoir is operated only once for each input during the training procedure. [14] Accordingly, this ease of training suggests that physical devices could be trained in the same way for executing neuromorphic computation. Using physical systems instead of neural networks has been previously defined as physical reservoir computing.[20]

Training hardware can be advantageous over training software models for several reasons. Training time of hardware can be very small compared to software models. In contrast to software models, the time needed to operate a piece of hardware is independent of the hardware size. Therefore, one can increase the size of hardware without severe increase in the training time. In summary, with physical reservoir computing, large pieces of hardware can be trained with big data relatively fast. This offers a possibility of developing an alternative way of neuromorphic computation more efficient than the nowadays widely used deep learning models.

1.2 Content of the Thesis

The text is organised as follows. In chapter 2, key mathematical primitives important for understanding reservoir computing are explained. These mathematical primitives feature frequently in the work and manifest themselves in many different forms.

In chapter 3, papers I, II and VII are summarised. These papers introduce experimental components that exhibit memristive behavior. These elements can be described by using a memristor like model. In papers I and II memristors are considered as physical elements. A memristor is a piece of hardware with a memory-like property. In these theoretical studies we investigate how they can be used to sense ionic concentrations. Paper VII is both a theoretical and an experimental study. In this paper, an oxytocin sensor (OT-sensor) and its equivalent model have been considered for experiments and simulations respectively. The OT-sensor has memory properties and its dependency on ionic concentrations has been found experimentally and published in literature [21].

In paper I, it is demonstrated theoretically on a very simple classification problem that a single memristor can be used to classify the environment it is exposed to. Only two environmental conditions have been considered, describing a static and a varying environment. Firstly, by an intuitive analysis, it has been shown that environment classification can be encoded in the memristor's state. A suitable drive signal has been suggested based on intuitive reasoning. Then, a rigorous mathematical optimization problem has been set up, and solved by using genetic algorithms. The optimization algorithm was allowed to search through a wider space of drive signals. While the "intuitive drive" was a square-wave like form, the optimization algorithm was allowed to search in the space of more complex sinusoidal drive signals.

In paper II, the cooperative behavior between memristor elements has been explored with a goal of achieving an additional functionality. In particular, it has been investigated how the interplay between various network features influences the sensing (computing) capacity of the device. The features of interest included the number of network elements, their connectivity pattern, and the complexity of the individual element. A big question which has been addressed is how to quantify in rigorous mathematical terms the sensing capacity of device.

In paper VII, an OT-sensor has been considered for sensing whether zinc ion concentrations are stable or varying in their surrounding aqueous environment. The findings of optimising an external drive signal to increase the sensing capacity, which have been shown in papers I and II, have been used here. However, in paper VII, it was hard to encode environment related information in instantaneous values of the OT-sensor's state. It was possible to encode this information in the history of sensor's state. In that case, a readout layer with memory of the sensor's state would be necessary.

In chapter 4, papers V, VI and VIII are summarised. Key findings from chapter 3 are exploited for developing generic theories on temporal information processing problems. In these three papers, memristor elements have been considered.

In paper V, a generic theory has been introduced for quantifying the computing capacity of a generic reservoir. This theory can be used to compare many reservoirs based on their ability to separate inputs. The key idea follows the central dogma of reservoir computing, *i.e.* that the best reservoir should separate any pair of input signals at the largest extent.

In paper VI, one memristor element has been considered for a prediction problem. The problem is to predict at an early stage whether a patient in intensive care unit has the sepsis by reading the history of the patient's measured medical variables. The system consists of a simple input layer, the memristor element and a memoryless output layer. Only the input and output layers are optimised. It has been shown that performance can be greatly improved by training those layers but cannot compete the performance of deep learning algorithms. In addition, in this thesis, a novel method is provided and shows that many memristor elements can be combined to achieve comparable performance to deep learning models. This method is generic and can be applied at other prediction problems as well.

In paper VIII, a novel method is proposed to reduce the number of reservoir elements but without reducing the reservoir's computing capacity. This method makes use of the previously referred findings regarding optimising an external drive signal. It is demonstrated on a problem of electrocardiogram signal classification that learning a drive signal can significantly improve the performance of a reservoir consisting of just a single memristor element. Similar ideas feature in machine learning as "feature engineering".

In chapter 6, papers III and IV are summarised. In those papers, new algorithms have been introduced for the efficient simulation of electronic circuits consisting of experimental elements.

In paper III, a new method has been introduced for the efficient simulation of electronic circuits which contain Constant Phase Elements (CPEs). We suggest an algorithm for simulating circuits with CPEs based on the Modified Nodal Analysis (MNA). The algorithmic complexity of the suggested algorithm is linear with the total time of the transient simulation. This algorithm is compared to a simple method found in the literature: the consideration of resistance-capacitor circuits with equivalent impedance to the CPEs. The comparison has been done both in terms of accuracy and algorithmic complexity.

Paper IV is a typical device modelling paper. A simple dynamical model of the Organic Electrochemical Transistor (OECT) element has been suggested and implemented for simulation purposes. The model has been systematically improved through a series of carefully designed numerical experiments. The key outcome of this work is a rigorous simulation algorithm that can be used to predict the response of the currents through the transistor's pins in time, depending on the history of voltages that are applied at its pins. A key challenge that has been addressed was to explain intriguing peaks in the experimental data for the drain current as a function of time.

Chapter 2

Mathematical Primitives

The filter is a mapping that converts an input series of data $q \equiv \{q(t)\}_{t\in I}$ into a state time series data $x \equiv \{x(t)\}_{t\in I}$,

$$q \xrightarrow{\mathcal{F}} x \tag{2.1}$$

where I denotes the index set and t denotes time. The individual values in the series are indexed by the variable t, *e.g.* as $q(t)$ or $x(t)$. In the following, the index set will be omitted when it is irrelevant for a discussion. The operation of the filter is denoted as

$$x = \mathcal{F}[q] \tag{2.2}$$

and a specific value indexed by t can be selected as $x(t) = \mathcal{F}[x](t)$. Further, the input and output data types do not need to match at all. For example, a filter $\mathcal{F}$ can take a vectorial data as the input, a series of values $\{(q(t), u(t)\}_t$ with $q, u \in \mathbf{R}$ and produce a single valued series $\{x(t)\}_t$ with $x \in \mathbf{R}^{\mathbf{N_R}}$ as the state, with $\mathbf{R}$ being the set of real numbers and N_R being the dimensionality of $x(t)$. For a given index t one has $x(t) = \mathcal{F}[q, u](t)$.

The reservoir is a special type of a filter. It is a generic dynamical system, to be denoted by $\mathcal{R}$, that responds to a time-dependent external signal $q(t)$ in a way that the state of the system x at a particular time instance t depends on the way the system has been driven in the past. Using the filter notation introduced above, this behavior is represented as

$$x(t) = \mathcal{R}[q](t) \tag{2.3}$$

In this case the index set I is meant to describe the flow of time. Note that the equation does not read $x(t) = \mathcal{R}(q(t))$, which would imply that the state of the system is an instantaneous function of $q(t)$.

A reservoir to have the filter property should consist of elements with memory properties. A model of such an element has been described in literature as the recurrent cell. [22] A recurrent cell is expressed mathematically as a model where the model's state at time t, $R(t)$, depends on the state of the short past $R(t - dt)$ (where $dt \to 0^+$) and on the model's input $u(t)$ through a non-linear function η:

$$R(t) = \eta\Big(R(t - dt), u(t)\Big) \tag{2.4}$$

Recurrent cells are mainly used in computer science to build recurrent neural networks (RNN) by inter-connecting them. In an RNN, the state of one interconnected recurrent cell may depend on the short past state of many other recurrent cells connected to it. RNNs can be used for pattern recognition tasks if the connections between the cells and the readout layer are trained for this purpose.

Reservoir computing is a sub-field of RNNs where the connections between the cells are not trained but just a readout layer is trained. This could be applied in cases

where it is not feasible to train connections between interconnected models. For example, in the case of executing pattern recognition tasks with physical systems, it may not be possible to intervene in the internal structure of the material and modify it. Additionally, it may be impractical in terms of time to train a model with a huge number of internal parameters, and training a model consisting of many internal parameters with a relatively big data-set could last weeks or even months. This long time of training could be avoided by using reservoir computing because it is not needed to train model's internal connections.

In reservoir computing, it is only needed to train the readout layer, to be denoted by ψ. This layer analyses the instantaneous state of the device x and produces the output y

$$y(t) = \psi(x(t)) \tag{2.5}$$

Note that the equation does not read $y(t) = \psi[x](t)$ which would imply that ψ represents a filter. Training the readout layer is important because one should be able to query the reservoir's state. Further, the apparatus used to query the state should be something simple, with a low degree of computational complexity, and presumably something that is easy to engineer. This is the reason why a common reservoir computing practice is to train a linear readout layer.

The key claim of reservoir computing: The abstract mathematical formulations introduced above formalize the reservoir computing ideas. Clearly, without stating what the expressive power of this model of computation is, the mathematical primitives are an empty shell without substance. What gives substance to the field is the claim that if the filter $\mathcal{R}$ has some well-defined mathematical properties, notably if it separates the input, then any computation is possible with one and the same reservoir $\mathcal{R}$. Thus for every desired pattern recognition task $\Phi[q](t)$, it is possible to find a related readout layer ψ_Φ such that

$$\Phi[q](t) = \psi_\Phi(\mathcal{R}[q](t)) \tag{2.6}$$

This implies that a single dynamical system has, in principle, infinite computing power, *i.e.* it can be used to compute anything. At first this might sound as an impossible claim, but in fact this key insight from the liquid state machine model rests on rigorous mathematical foundations of the Stone Weierstrass Approximation theorem. [12].

The sensing goal: Every sensing procedure is done with a certain goal in mind. For example, one might be interested in inferring whether a solution containing ions is static or changes in time. Thus a sensing procedure can be formally described as a pattern recognition task, described by the filter Φ,

$$\varphi(t) = \Phi[q](t) \tag{2.7}$$

The filter is constructed so that its output, the variable $\varphi(t)$, convey the pattern recognition information. For example, the filter could be constructed to output $\varphi \approx 0$ for static ion concentration, or $\varphi \approx 1$ for a varying one. It is useful to think of Φ as an infinitely "intelligent" neural network that can be trained for any pattern recognition task.

The sensing reservoir is defined a special dynamical system that can be driven by an external input u and interacts with the environment q. An implementation of the sensing reservoir idea is shown in Fig. 2.1 where a reservoir is assumed to be embedded in an ionic solution. A network is used as a dynamical system, the reservoir $\mathcal{R}$. The network consists of N_R interconnected electronic elements R_1, R_2,

$\cdots$, R_{N_R} in an electronic circuit, where $N_R = 3$ in Fig. 2.1. Those elements are assumed to behave as recurrent cells, where their state at time t, $R_m(t)$ depends on the state of the short past $R_m(t - \Delta t)$, their individual voltage differences, $u_m(t)$ and the environmental condition $q(t)$:

$$R_m(t) = \eta_m\Big(R_m(t - dt), u_m(t), q(t)\Big) \tag{2.8}$$

By analysing the above equation, the state $R_m(t)$ can also depend on the states $R_j(t - dt), m \neq j$ if the individual voltage difference $u_m(t)$ depends on $R_j(t - dt)$. This is possible, if the electronic components are connected into a network. Then, the state of one electronic component would depend on the history of the states of the other electronic components:

$$R_m(t) = \eta_{mR}\Big(R_1(t - dt), R_2(t - dt), \cdots, R_{N_R}(t - dt), u(t), q(t)\Big) \tag{2.9}$$

A network consisting of such electronic components implements the filter primitive since each element behaves as a recurrent cell. The state of the network at time t, $x(t)$, is given as:

$$x(t) \equiv (R_1(t), R_2(t), \cdots, R_{N_R}(t)) \tag{2.10}$$

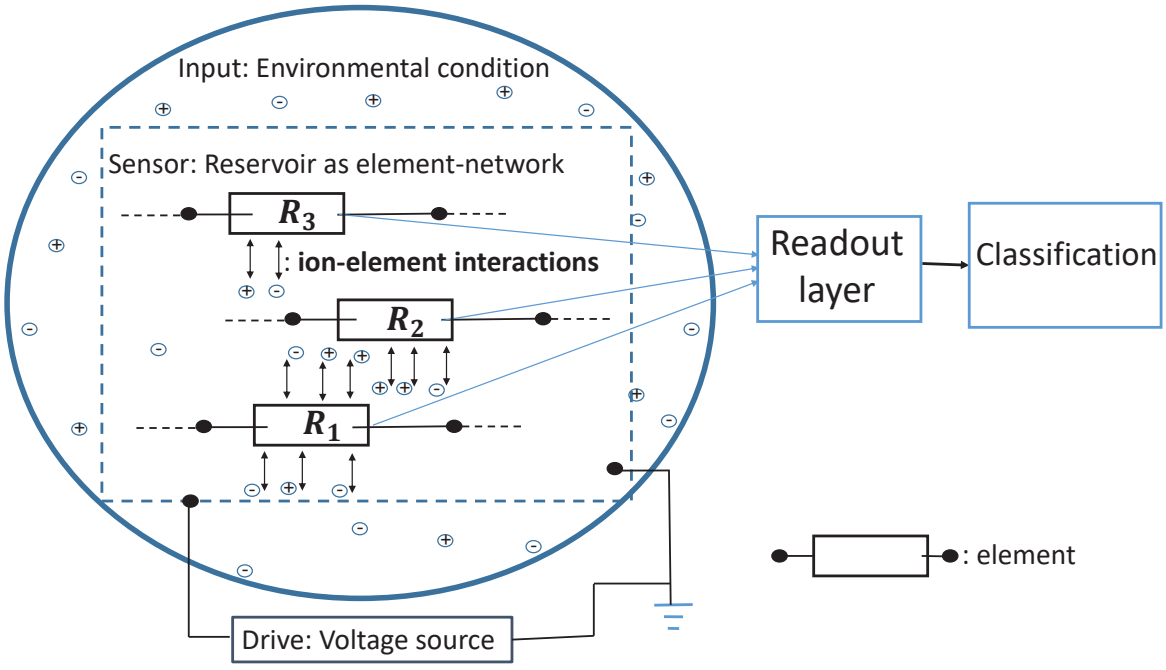


FIGURE 2.1: A modification of figure taken from paper I. The network is driven by a signal and is affected by the environmental conditions of ionic solutions. The readout layer receives the instantaneous values of the network state and contributes to the classification of the environmental condition.

In the filter notation, the state of the reservoir can be written as

$$x(t) = \mathcal{R}[u, q](t) \tag{2.11}$$

and, the sensing performed by the reservoir is represented as

$$y(t) = \mathcal{S}[u, q](t) = \psi(x(t)) \tag{2.12}$$

where $y(t)$ is the variable that conveys the result of the sensing measurement. *The key idea is that all the sensing functionality should be done by the reservoir, and not the readout layer.* In principle, in reservoir computing, it is not aimed at identifying ways of finding a readout layer which would improve the sensing performance. It is aimed at finding ways which would improve only the functionality of the reservoir. Thus the readout layer should be something simple to engineer with a low degree of computational complexity such as a linear readout layer:

$$y(t) = w_0 + w_1\, R_1(t) + w_2\, R_2(t) + \cdots + w_{N_R}\, R_{N_R}(t) \tag{2.13}$$

where $W = (w_0, w_1, \cdots, w_{N_R})$ are linear weights of the readout layer. In this thesis, whenever a linear readout layer is considered, the linear weights are calculated with least square error regression.

Since the readout layer is a simple structure, accurate sensing may not be possible if the reservoir were not complex enough. In physical reservoir computing there may not be an option to intervene, modify and increase the reservoir complexity, where, by assumption, one is interested in building computers from dynamical systems that cannot be easily modified. [23, 24] In this thesis, it is suggested that it is possible to work around this problem. The key idea is that an external drive signal can be optimised to achieve advantageous correlations between environment and reservoir's state, increasing the performance of reservoir computing.

A hand drawn illustration of the sensing reservoir concept (a modification of a figure from [18]) is shown in Fig. 2.2. The reservoir is denoted by $\mathcal{R}$ with the state of the reservoir at time t denoted by $x(t)$. This state depends on the whole history of the drive signal u and the environmental condition signal q. To optimize the sensor, a drive signal u has to be found such that the output y is driven to 1 if the environmental condition is a varying one, and to 0 if the environmental condition can be characterized as a stable one.

Sensor optimization: The goal is to optimize the sensing reservoir so that it mimics the behavior of Φ: Formally, one tries to achieve that $y(t) \approx \varphi(t)$ to the largest extent possible, uniformly over time. This defines a rigorous mathematical optimization problem, where the goal is to find a drive u_* such that

$$u_* = \text{argmin}_u \delta[u, q] \tag{2.14}$$

where $\delta[u, q]$ is a measure of how well the prediction of the sensing reservoir matches the desired classification goal Φ,

$$\delta_\Phi[u, q] \equiv ||\mathcal{S}[u, q](t) - \Phi[q](t)||_t \tag{2.15}$$

with $||...||_t$ being a measure of the distance between two filters, the one realized by the reservoir $\mathcal{R}$ and the one by Φ. In the following, when the sensing goal is known or not central for the discussion instead of $\delta_\Phi[u, q]$ we write $\delta[u, q]$, omitting the explicit dependence on the sensing goal Φ. The subscript on the distance symbol indicates that one should in some sense provide a distance estimate over all times. For example, one could define the distance as

$$||\mathcal{S}[u, q](t) - \Phi[q](t)||_t^2 \equiv \lim_{T\to\infty} \frac{1}{T} \int_0^T dt\, \{\mathcal{S}[u, q](t) - \Phi[q](t)\}^2 \tag{2.16}$$

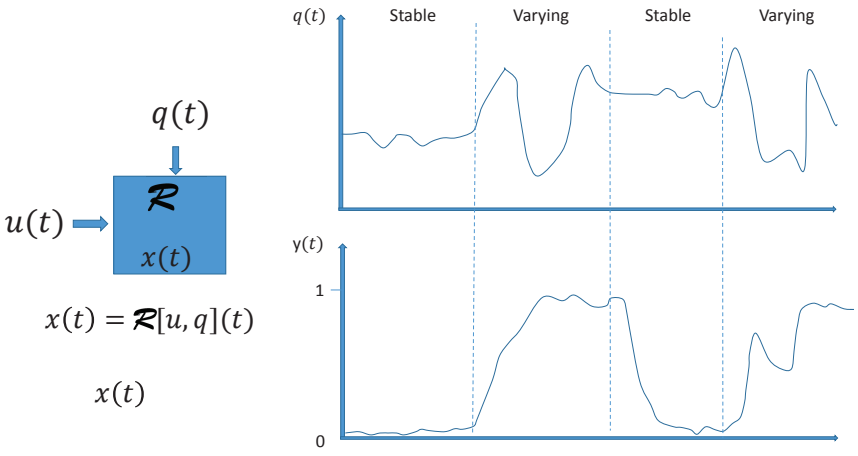


FIGURE 2.2: A hand drawn illustration of the sensing reservoir concept (a modification of a figure from [18]).

which is the definition used in the thesis. The distance measure generated this way will be denoted as $\delta^*[u, q]$.

Chapter 3

On using reservoir computing for developing sensing applications

How does one optimize a reservoir for a particular pattern recognition task? One of the recurring challenges in this thesis is to, for a given network, find a drive signal u_* that maximizes the sensing capacity of the network $SC[u, q]$:

$$u_* = \text{argmax}_u SC[u, q] \tag{3.1}$$

It is hard to define the sensing capacity rigorously but not impossible. For example, for a given sensing goal Φ one can define the sensing capacity specific to this sensing goal by using the prediction errors $\delta_\Phi[u, q_i]$, $i = 1, 2, \cdots, E$. In machine learning one measures the total prediction error,

$$\epsilon = \sum_{i=1}^{E} \delta_\Phi[u, q_i] \tag{3.2}$$

and a viable definition of the sensing capacity measure would be

$$SC_\Phi[u] \sim \frac{1}{\epsilon} \tag{3.3}$$

indicating that a small prediction error should be associated with a large fitness. However, a slightly different estimate could be used

$$SC_\Phi[u] \sim \sum_{i=1}^{E} \frac{1}{\delta_\Phi[u, q_i]} \tag{3.4}$$

Finally, the overall sensing capacity of the device can be calculated as the average over the individual sensing capacities

$$SC[u] \sim \langle \delta_\Phi[u, q_i] \rangle_\Phi \tag{3.5}$$

where the symbol $\langle ... \rangle_\Phi$ denotes the averaging procedure.

However, such a definition of the sensing capacity is dependent on how the averaging procedure $\langle ... \rangle_\Phi$ is performed and ultimately which classification problems are covered. Instead, another definition of the sensing capacity is provided in the thesis that does not rely on how one defined the classification goals, but only depends on how the sensor behaves in general. The key assumption in these studies is that the trajectory separation in the phase space controls the sensing capacity.

Mathematically, trajectory separation can be described as follows. Let $q_1, q_2, \cdots, q_E$ denote the set of distinct environmental conditions to be classified. If the state of

the reservoir occupies different regions $\Omega_1, \Omega_2, \cdots, \Omega_E$, when exposed to the conditions $q_1, q_2, \cdots, q_E$ then a classification is possible. In particular, if the separation of the trajectories is strong, the classification could be achieved with a relatively simple memoryless readout layer, *e.g.* a linear classifier might be sufficient as the one in Eq. (2.13). Otherwise, if separation of trajectories is not strong, then, a more complex readout layer is needed, probably a readout layer with memory and non-linear properties.

The sensing capacity $SC[u, q]$ is an overarching concept that features in many papers and naturally generalizes to the notion of computing capacity. In fact, it is often hard to distinguish the two, the distinction essentially being defined by the application context. The exact definition of the sensing capacity $SC[u, q]$ varies in different papers depending on the needs. Typical papers that deal with the sensing/computing capacity topic are in papers I, II, VII and VIII.

3.1 Sensing with memoryless readout layers

In papers I and II memristor models have been considered as recurrent cells of reservoirs. In those papers, the purpose has been to define sensing capacity measures which correlate with the degree of trajectory separation. Those measures were used as fitness functions when training memristor networks. The purpose was to find a drive signal u so that the degree of trajectory separation is large. In those papers, a large number of memristor networks were tested and relatively large sensing capacities were found for some of them.

A memristor is a non-linear, passive, two-terminal component with a time-varying resistance often being referred to as the memristance $R(t)$. The memristor element is suitable for temporal information processing since it exhibits the filter property: The memristance value at a specific time instance depends on the whole history of the applied voltage signal up to that time.

In this thesis, a simple Pershin Di Ventra model [25] is used. The memristance $R(t)$ changes depending on the voltage signal ΔV that is applied across the element according to a simple law. If $-V_{thr} < \Delta V < V_{thr}$, then the memristance changes as $\dot{R} = \alpha \Delta V$, where here and in the following the dot over a symbol defines a time derivative $\dot{x} = \frac{dx}{dt}$. For $\Delta V < -V_{thr}$ or $\Delta V > V_{thr}$, $\dot{R} = \beta \Delta V + const$; α, β are device dependent parameters, $const$ is a constant value and usually $\alpha \ll \beta$. The memristance is bounded between the lowest value $R_{min} > 0$ and the maximum value R_{max}. This can be written as:

$$\dot{R}(t) = f(\Delta V(t), \beta)\Theta(R(t), \Delta V(t)) \tag{3.6}$$

with

$$f(\Delta V, \beta) = \beta \Delta V + \frac{1}{2}(\alpha - \beta) \cdot \left(|\Delta V + V_{thr}| - |\Delta V - V_{thr}|\right) \tag{3.7}$$

and

$$\Theta(R, \Delta V) = \begin{cases} 0, & \text{if } \Delta V = 0 \\ \theta(R_{max} > R), & \text{if} \Delta V > 0 \\ \theta(R > R_{min}), & \text{if} \Delta V < 0 \end{cases}$$

where $\theta(R_{max} > R)$ is zero unless the condition in the argument is satisfied, and likewise for $\theta(R > R_{min})$.

The environment model: By assumption, the network is affected by the environmental conditions of the ionic solutions surrounding it. The key challenge is to assume a suitable model for the environment-memristor interaction. There are numerous options, and special care has been given to choosing an appropriate model. In paper I, based on a careful literature study, it has been argued that it is reasonable to assume that the rate of the memristance change should depend on the ion-concentration. Thus, for simulation purposes, it has been assumed that the parameter β is environment sensitive. Assuming that the variations of the environmental signal are small, one can use the standard working point model used in electronics: $\beta(t) = a + bq(t)$.

3.1.1 Sensing with one-memristor reservoirs

In paper I, the simplest possible network, with one element, has been trained to handle a classification problem with two environments. Since the emphasis is on testing the overall workings of the method, a relatively simple classification problem has been chosen. The goal is to distinguish between two different environments, a stable and a varying one. These were represented by a relatively simple signal pair denoted by q_1 and q_2 and shown in Fig. 3.1. The figure has been taken from paper I. Normally, in the supervised learning approach, a class is represented by a group of similar signals, but we have considered only one signal per class. In such a way it is possible to have an intuitive understanding how the optimal drive should look like.

As discussed in chapter 2, this classification problem can be represented as an optimization problem where the goal is to minimize the distances between the classification performed by the system and the desired classification. The goal is to train the system, *i.e.* to find the drive u_*, so that sensing is done as

$$S[u_*, q_1] \approx \Phi[q_1] \tag{3.8}$$
$$S[u_*, q_2] \approx \Phi[q_2] \tag{3.9}$$

where the pattern recognition problem that needs to be learned by the memristor is given by $\Phi[q_1](t) = 0$ and $\Phi[q_2](t) = 1$.

The above optimization problem has been solved using genetic algorithms, where the following fitness function was used as the optimization goal:

$$SC_1[u] = \sum_q \frac{1}{\delta^*[u, q]} \tag{3.10}$$

The sum is over environmental conditions. Effectively, the fitness function above, describes the fact that the goal is to find the smallest possible distances for every environmental condition. If both distances are small, SC_1 is large. In a way, the fitness function specified above is a measure of the sensing capacity of the system.

A drive signal was found firstly by direct intuitive reasoning. The dynamics of the memristor element were analyzed and there was a good understanding of how to choose a drive signal to lead the memristance to a specific direction under one specific environmental condition. For the two environmental conditions, a drive signal was identified so that the memristance occupies two different regions under the two environmental conditions respectively. This is shown in Fig. 3.2 where the output is shown under the environments q_1 and q_2 and the found drive signal. The intuitive reasoning was based on the fact that a drive signal can be found such that the memristance increases and decreases with the same rate under the stable environment q_1

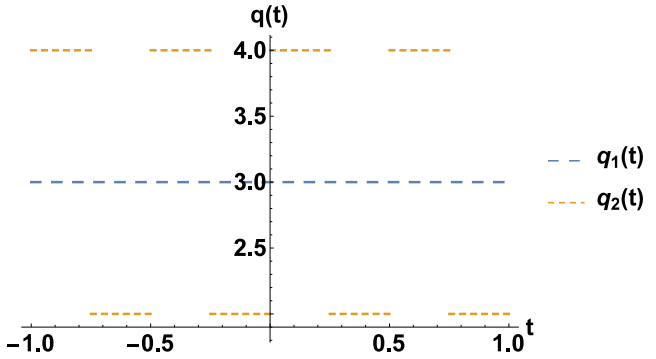


FIGURE 3.1: Figure taken from paper I. The environmental conditions which are denoted as q_1 and q_2.

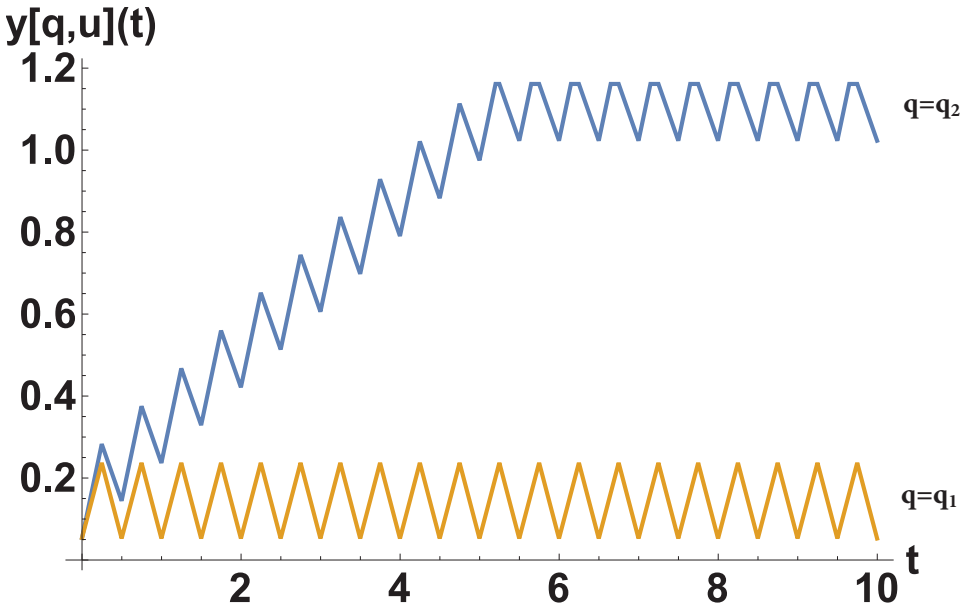


FIGURE 3.2: Figure taken from paper I. The simulated outputs of the one-memristor network under the intuitively found drive signal and the environmental conditions q_1, q_2.

resulting in stable memristance around the initial value. Additionally, for the same drive signal, under the environmental condition q_2, the memristance increased with a larger rate than decreased. This resulted in a constantly increasing memristance. Therefore, the memristance occupied different regions under the conditions q_1 and q_2.

This intuitive solution was found to be in a good agreement with the drive obtained by running the genetic algorithm. The output for this drive signal and each of the environmental conditions q_1 and q_2 is shown in Fig. 3.3. Under the condition q_1 and the optimized drive signal, the dynamics were such so as the memristance decreases on average. However, under the condition q_2 and the drive signal u_*, the memristance increased on average. Interestingly, under the optimized drive the output is driven faster to the target values $\Phi[q_1](t)$ and $\Phi[q_2](t)$ than the output for the intuitive drive (Fig. 3.2). However, there is a larger variance around the target values with the optimised drive than the intuitive drive.

There are mainly two major results by finding a drive signal with both ways.

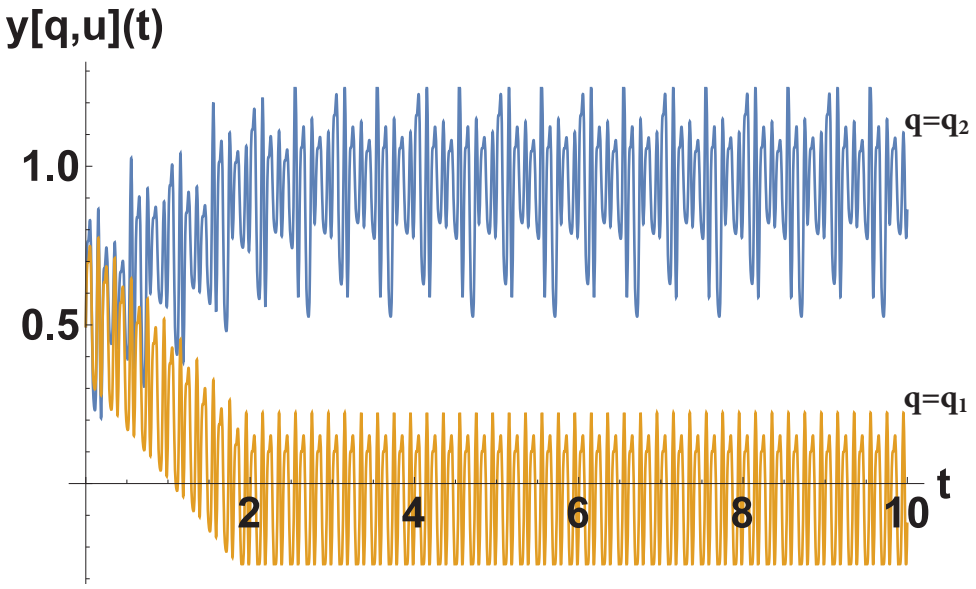


FIGURE 3.3: Figure taken from paper I. The simulated outputs of the one-memristor network under the optimum drive signal (from the genetic algorithm) and the environmental conditions q_1 and q_2.

Firstly, the synchronization between the drive signal and the environmental conditions is important so that memristance can be driven to different regions under different environmental conditions. The phase space separability idea is illustrated nicely in Figs. 3.2 and 3.3. However, the space separability deteriorates by a slight loss of synchronisation. This is shown in paper I by implementing numerical experiments with slightly modified either the optimised drive signal or the signal q_2. Secondly, several numerical simulations were implemented and it was found that when increasing the amplitude of the drive signal, there was a faster response but a larger variation around the target $\Phi[q]$. This indicates that there is a trade-off between time response and variation around the target $\Phi[q]$.

3.1.2 Towards collaborative sensing

In paper I, it was found that a one-memristor network can be used for classifying two environment time-series q_1 and q_2. This indicates that a many-memristor network, with many different memristors, could be used to separate between different features of the environmental conditions, and perform a generic classification. This has been investigated in paper II where the classification task is the same as in paper I, *i.e.* the network should distinguish between a varying or a stable environment. To test how well the system generalizes, a larger number of environmental conditions is considered. Every environment q_i is represented as group of signals $q_i \equiv \{q_i^a; a \in E_i\}$ where E_i is the index set that describes the environment q_i.

Simulations were done to investigate a specific idea. One memristor of the network could contribute to the separation based on different features between a constant environment q_1^a and a varying q_2^a. Another memristor could contribute on the separation based on other different features between a constant environment q_1^b and a varying environment q_2^b. So on, other memristors could be used for separating other different features. For this to be done, all the memristor elements ought to be connected so that the separation of different features can be distributed among the

network elements. In other words, this would require the collaboration between the elements for which separation tasks they will be assigned.

Assuming that collaboration between memristor elements can be exploited as advocated, how should the elements be connected into a network for the best possible effect? In paper II, it has been attempted to answer this very broad question in three directions. Firstly, it has been investigated how the number of identical memristor elements and their connectivity patterns affect the sensing capacity of the network. Secondly, it has been investigated if it is possible to increase the sensing capacity by increasing the complexity of memristors. The complexity of memristors has been increased by adding heterogeneous time delay feedback mechanisms. Even though memristors are identical, they are allowed to have a different time delay feedback. Thirdly, it has been investigated if it is possible to increase the sensing capacity by adding heterogeneous memristors in a network but without a time delay feedback. In practice, there is variability across memristor elements. [26, 27] This means that in practice it is almost impossible to build two identical memristors. Therefore, in this direction, it is investigated whether memristor variability can be exploited for sensing.

To investigate network structures in these three directions, it is firstly needed to address the big questions of paper II:

- How can one quantify the sensing capacity of the reservoir without considering the readout layer? An equivalent question is, how much information about the environment can be stored in the state of the reservoir?
- Is the sensing capacity necessarily favored by just increasing the number of memristor elements?
- Which connectivity patterns are favorable for a larger sensing capacity? In particular, can delay feedback mechanisms be used with an advantage?

The time series of the environmental conditions (the training data) are shown in Fig. 3.4. These signals will be called the training data since the drive signal has been optimized for those conditions. The training data are labeled with their corresponding class and therefore finding the optimum drive signal is a supervised learning task. In order to test whether the optimum network can be generally used for classifying stable and varying environmental conditions, a bigger labeled data set was created, the testing data. The testing data consists of thousand randomly created conditions accompanied with the label of their corresponding class.

In paper II, a readout layer has not been considered because optimising the readout layer should not affect the optimization process, or influence the conclusions. An attempt has been made to focus exclusively on the sensing capacity of the reservoir per se. In papers I and II, the sensing capacity of the network was based on measuring the interclass separability, *i.e.* when the network is driven by different environmental conditions, then the state should be driven to different regions.

One of the questions that was needed to answer in paper II was how to quantify the sensing capacity of a memristor network without using a predefined readout layer. In paper II, a measure has been used which is indicative of the state separation, the separability index ν. The concept behind the definition of the index ν is illustrated in that each memristance of the network should be driven on average to different regions when different environmental conditions are applied. The state should be linearly separable on average.

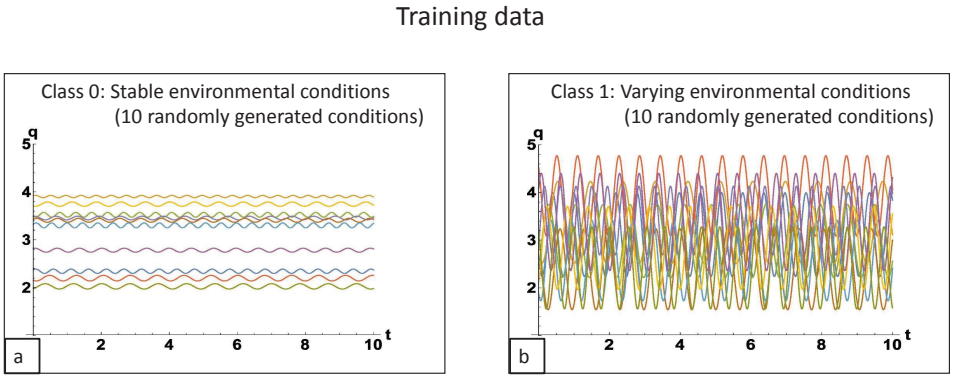


FIGURE 3.4: Figure taken from paper II. The training data for the two different classes. a) The training data for Class 0 b) The training data for class 1.

Mathematically, the ideas discussed above were implemented as follows. The mean value of the m^{th} resistance of a memristor network under the environmental condition q_i^a is given as:

$$\bar{R}_m[u, q_i^a] = \frac{1}{T}\int_0^T dt R_m[u, q_j^i](t) \tag{3.11}$$

For a given network with N_R memristors and under a drive signal u, the distance between two environmental conditions belonging to classes i and j, q_i^a and q_j^b, is given as:

$$d_{j,b}^{i,a} = \sqrt{\frac{1}{N_R}\sum_{m=1}^{N_R}\left(\bar{R}_m[u, q_i^a] - \bar{R}_m[u, q_j^b]\right)^2} \tag{3.12}$$

Totally, for a given network, a drive signal u and a set of training data with classes $c_1, c_2, \cdots, c_k$, the index ν is calculated as the geometric mean of all the possible N_D distances $d_{j,b}^{i,a}$

$$\nu[u; c_1, c_2, \cdots, c_k] = \left(\prod_{i=1}^{k}\prod_{i'=i+1}^{k}\prod_{j=1}^{N_i}\prod_{j'=1}^{N_{i'}} d_{j,j'}^{i,i'}\right)^{\frac{1}{N_D}} \tag{3.13}$$

The structures of the memristor networks were investigated in three directions. In the first direction, the complexity of the network topology was increased by adding identical memristors in parallel and in series. Six networks were considered: $N1, N2, \cdots, N6$ with the number of memristors given by $N_R = 1, N_R = 2, \cdots, N_R = 6$ respectively. In the second direction, the complexity of the elements was increased by introducing the MFB element: a memristor with a time-delay feedback loops. Time-delay feedback loops were used because the system is expected to gain additional memory properties. The MFB unit at the time t keeps track of the memristance of a previous time with a delay τ, $R(t-\tau)$, and converts it to a voltage signal with a linear mapping. The converted voltage is added to the voltage signal across the memristor element. An example is given in Fig. 5 of paper II which shows two networks the $NFB1$ and the $NFB2$ with 1 and 2 MFB units respectively. In the network $NFB2$ an MFB unit is added to the network $NFB1$: The memristance signal of $NFB1$ is converted to voltage and is used to drive the added MFB unit. Similarly,

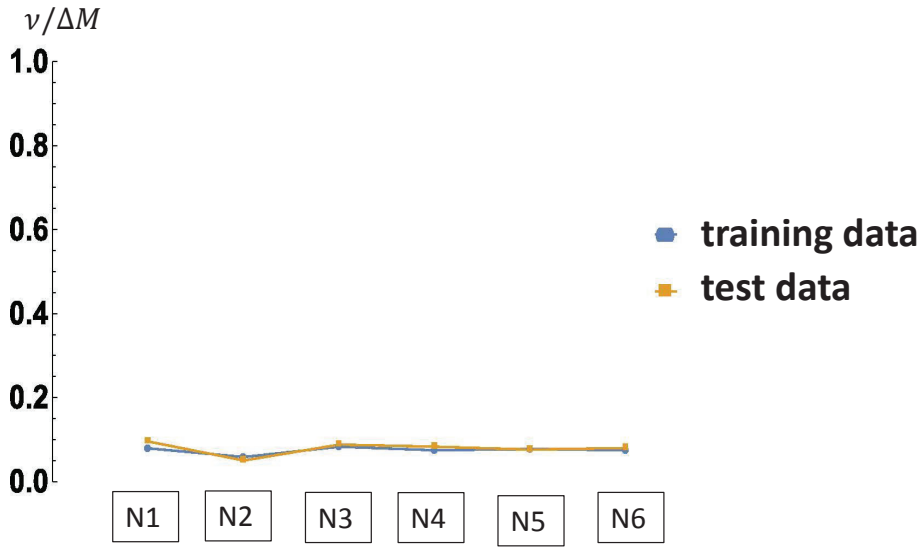


FIGURE 3.5: Figure taken from paper II. The optimum separability index when considering the training data and the measured separability index on the test data for the different network structures $N1$, $N2$, $N3$, $N4$, $N5$ and $N6$.

the network $NFB3$ was considered by adding an MFB unit to $NFB2$, the network $NFB4$ by adding an MFB unit to $NFB3$. In a similar way, $NFB5$ and $NFB6$ networks were considered. For the networks $NFB2$, $NFB3$, $\cdots$, $NFB6$, the time delays of the MFB units were also considered as free parameters to be optimized. In the third direction, heterogeneous memristor elements were added in series and in parallel. It has been assumed that the parameter $\beta(t)$ of each memristor depends differently on the environmental condition $q(t)$:

$$\beta(t) = (1 + m_0)q(t) + m_0 \tag{3.14}$$

where m_0 is assumed to be unique for every memristor element and randomly chosen between 0 and 1. In Fig. 6 of paper II, the networks constructed in the third direction $NSP2$, $NSP4$, $NSP6$ and $NSP8$ are shown.

The separability index ν was maximized by training all the memristor networks with the training data. Additionally, to evaluate the performance of the optimized networks, the separability index was measured on the test data. Regarding the first direction, the index ν for training and testing the networks $N1$, $N2$, $\cdots$, $N6$ is shown in Fig. 3.5. Regarding the second direction, the index ν for training and testing the networks $N1$, $NFB1$, $NFB2$, $\cdots$, $NFB6$ is shown in Fig. 3.6. Regarding the third direction, the index ν for training and testing the networks $NSP2$, $NSP4$, $NSP6$ and $NSP8$ is shown in Fig. 3.7.

In the first direction, the separability index did not increase when adding memristor elements in parallel and in series. With such network structures and by adding identical memristor elements there is no adequate collaboration between the memristor elements.

In the second direction, the separability index increased when adding memristor elements with a time delay feedback. More specifically, as the dimension of the state increases from ($N_R = 1$ for $N1$) towards ($N_R = 4$ for $NFB4$), then, the separability index on both training and test data increased. This means that elements collaborate

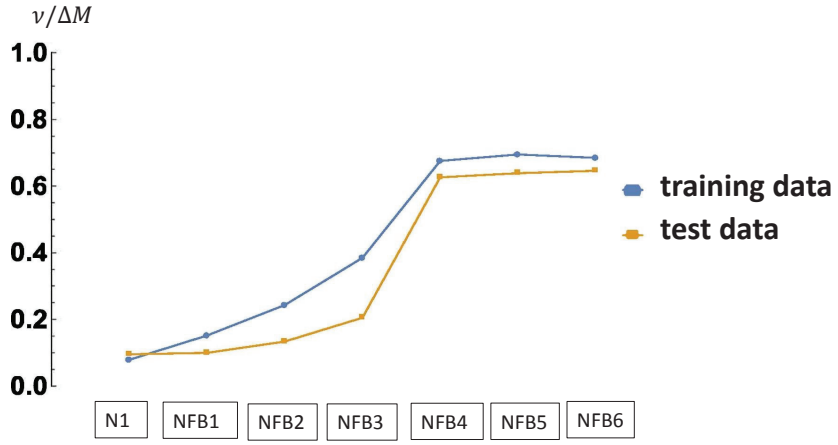


FIGURE 3.6: Figure taken from paper II. The optimum separability index when considering the training data and the measured separability index on the test data for the different network structures $N1$, $NFB1$, $NFB2$, $NFB3$, $NFB4$, $NFB5$ and $NFB6$.

when added into the network. Additionally, the index ν did not improve considerably for both the training and the test data when adding MFB units to $NFB4$. Therefore, $NFB4$ can be considered as the network with the minimum amount of resources for achieving the largest index ν. One can also notice that the index ν on the testing data is favored by an increased number of MFB units. Especially, the distance between the index ν on training and testing data tends to be very small when considering the network $NFB6$. This happens because when the dimensionality of the state is large, then, there are more chances for the state to occupy different regions for different environmental conditions (phase space separation).

In the third direction, the separability index for the training data always increased when adding more memristor elements. This means that by increasing the dimensionality of memristor networks in the third direction, the reservoir could separate better the environmental conditions of the training data. However, the same did not happen for test data. The separability index for the test data is almost the same for both NSP2 and NSP8. This means that a drive signal was found for separating environmental conditions of the training data but not necessarily of the test data, a problem which is usually called as over-fitting in machine learning. To overcome over-fitting, one could train the drive signal on a larger number of environmental conditions in the training data at the cost of a larger training time.

3.2 Sensing with recurrent readout layers

So far, in the previous section, for some reservoirs, a drive signal has been found so that the reservoir's state is driven to different regions of the configuration space when the reservoir is exposed to different environmental conditions. However, there might be cases where this is hard to achieve. The state of the reservoir might be driven to overlapping regions of the configuration space under different environmental conditions. How should one deal with cases when it is impossible to find a drive signal for achieving a large degree of trajectory separation? Is it possible to find a drive signal so that information about the environment is encoded in the shape of trajectory time-series?

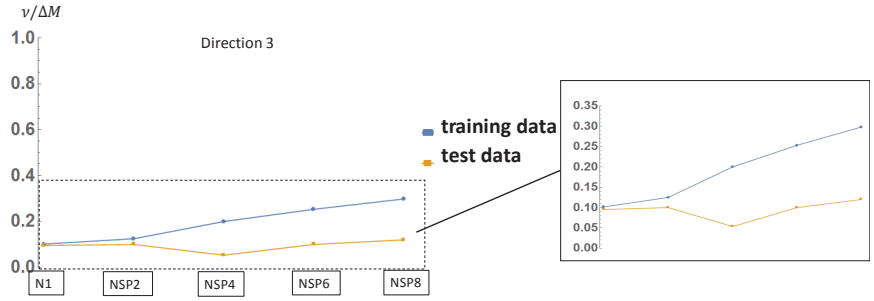


FIGURE 3.7: Figure taken from paper II. The optimum separability index when considering the training data and the measured separability index on the test data for the different network structures $N1$, $NSP2$, $NSP4$, $NSP6$ and $NSP8$.

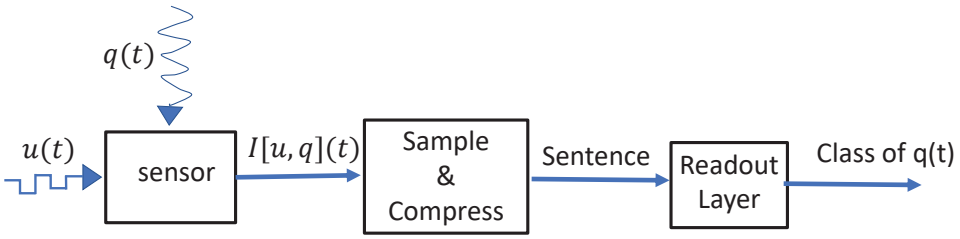


FIGURE 3.8: The proposed sensing setup with a recurrent readout layer.

These questions have been addressed in paper VII where a model of an existing biosensor, [21], has been considered as a recurrent cell. With this model, it was not possible to find a drive signal for achieving a large degree of trajectory separation. In paper VII, a method has been developed for encoding environment related information in the history of the state trajectory time-series instead of their instantaneous values. Then, to decode information about the environment, it is necessary to construct a readout layer with an access to earlier states of the device. Thus one needs a readout layer with memory. Then the information about the environment being encoded in the history of the state trajectory time-series can be processed. In this thesis, readout layers with memory properties are called *recurrent readout layers*.

3.2.1 Sensing with temporally extended bar codes

In paper VII, a method has been developed for encoding information about ionic variations in the shapes of the sensor's state trajectory signals. The purpose has been to train a drive signal so that the sensor's state trajectory signals have different shapes when the sensor is exposed to different environmental conditions.

The sensing setup is shown in Fig. 3.8. The sensor is exposed to a drive signal $u(t)$ and an environment signal $q(t)$. The sensor produces the state signal $I[u,q](t)$ as output. Then, the state signal is sampled and compressed by the next unit of the setup. This unit receives as input the state signal $I[u,q](t)$ and produces a sequence of characters, a sentence. The readout layer receives this sentence as input and produces as output the result of environment classification.

The proposed method is demonstrated in both theoretical and experimental terms by considering an Oxytocin monolayer based impedimetric biosensor for zinc and copper ions, to be referred to as the OT sensor. The equivalent electronic circuit

model for the OT sensor has been published in [21]. The OT sensor-environment model is obtained because the resistances of the equivalent circuit depend on the level of zinc and copper ionic concentration. Curves of those dependencies are provided by Tadi et al. in [21]. To demonstrate the method principles, the exposure of the OT sensor only to zinc ions has been considered, and a simple task has been investigated to infer whether the OT sensor is exposed to a stable or a varying zinc concentration. In paper VII, the equivalent circuit is considered only for small frequencies of the drive signal which is the voltage signal across the OT sensor.

The OT sensor's state is the current which flows across the OT sensor $I[q,u](t)$. This sensor has filter properties because the electronic equivalent circuit which describes the dynamics of the sensor contains constant phase elements. The voltage across constant phase elements at time t depends on the whole history of the current which has passed through the element.

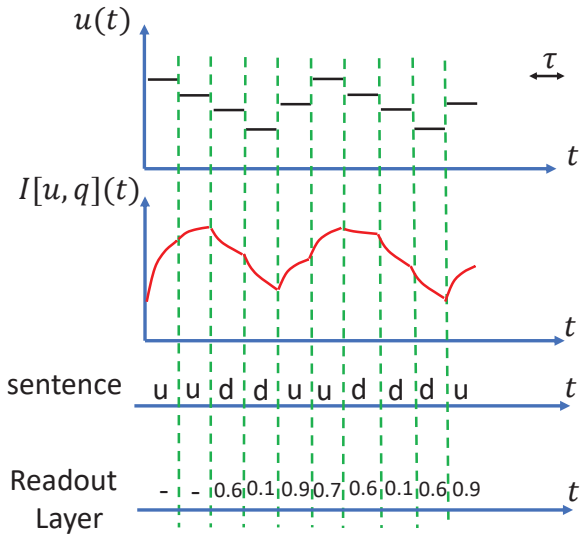
One problem with numerically simulating an electronic circuit with such elements is that the algorithmic complexity is $O(n^2)$ with n being the size of the time grid. In paper III, a method is suggested, tested and evaluated for reducing the algorithmic complexity by one order of magnitude from $O(n^2)$ to $O(n)$. This method has been used in paper VII for implementing numerical simulations of electronic circuits with the OT sensor. This method is summarised in chapter 5 of this thesis.

The sample and compress unit samples the output $I[u,q](t)$ and compresses it in a string of characters. One could compare temporally extended signals of the output $I[u,q](t)$ in order to extract environment related information. However, to do so, computationally expensive algorithms would be needed such as the cross-correlation method. To avoid such large computational costs, the output signal $I[u,q](t)$ is compressed into a string of characters. Then the task of the readout layer is to classify the environmental condition with specific probabilities.

A hand-drawn example of the sensing setup is shown in Fig. 3.9. The OT sensor is exposed to an environmental condition of zinc ionic variation $q(t)$ which is not shown in this figure. This means that the dynamics of the OT sensor depends on $q(t)$, *i.e.* the resistances of the equivalent electronic circuit model depend on $q(t)$. The drive signal $u(t)$ is chosen to be a periodical squared wave pulse, a very natural choice for this type of a device. Each period consists of five plateaus and each plateau lasts τ seconds. During each pulse, the current $I[q,u](t)$ relaxes towards stationary conditions with a relaxation time τ_*. This relaxation time depends on the resistances of the equivalent circuit which in turn depend on the levels of zinc concentration ($q(t)$). Therefore, the relaxation time varies when the zinc concentration varies. During each pulse, the sample and compress unit produces a character. In paper VII, this unit generates a character 'u' ('d') if $I[u,q](t)$ increases (decreases) on average during a pulse. The readout layer reads the sentence and remembers $n_c = 3$ characters to make a decision. For every three character word, the readout layer outputs the probability that the environment belongs to class 0 by finding this word in the readout layer's database. The database contains the possibility for the environment to belong to class 0 for every three character word.

In paper VII, the drive signal $u(t)$ and the database of the readout layer are trained on a set of environment signals. Here, the classification task is similar as in the previous section, *i.e.* to classify whether the environment (zinc concentration) varies or is stable. An advantage of the proposed method is that the training procedure can take place in two phases. In the first phase, the drive signal is trained and in the second phase the database of the readout layer is trained.

In the first phase, a drive signal is trained by supervised learning so that the



Readout Layer ($n_c = 3$)

WORD	Probability of Class 0
uuu	0.2
uud	0.6
udu	0.3
udd	0.1
duu	0.7
dud	0.8
ddu	0.9
ddd	0.6

FIGURE 3.9: A hand-drawn example of the sensing setup with a recurrent readout layer.

setup produces different character sequences under different environmental conditions. The training data consists of signals describing different patterns of ionic variations. The optimisation problem of finding a drive signal should be formulated so that the more different sentences from different environmental conditions are, the better a drive signal is and the larger its fitness is. If sentences coming from different environmental conditions are very different to each other, then, there are increased chances that specific words can be found that occur with a high probability only under one specific environmental condition. Such words, are called bar codes in paper VII.

The fitness of a drive signal can be evaluated by making comparisons between all the pairs of sentences which occur under different environmental conditions. For this purpose, a similarity measure, $\sigma[s_1, s_2]$ between any two sentences s_1 and s_2, is introduced in paper VII to estimate how similar two sentences are. For given classes of environmental signals, any two sentences associated with any two distinct classes of environmental conditions should be as different as possible. Likewise, the sentences associated with the same class should be as similar as possible. The similarity measure is quantified by using the concept of sequence alignment inspired by the field of bioinformatics. The well-known Needleman - Wunsch algorithm has been used, which has been published in [28]. A detailed implementation of this algorithm is given in paper VII.

In the second phase, the trained drive signal can be used to learn the database of the readout layer. The sensor is operated with the optimised drive signal and under many signals of the environmental condition $q(t)$. Then, statistics are implemented to update the database with the probability of a word to occur under each class. For example, if the word "udududuuddud" is found to occur 990 times under environment signals of class 0 and 10 times under environment signals of class 1, then, whenever the readout layer receives this word, it would infer that the environment belongs to class 0 with probability 99%.

Training the drive signal and the readout layer in two phases is advantageous for two reasons. Firstly, data sets with different sizes can be used for each phase. In particular, a training data set with a small size can be used in the first phase because learning the drive signal is computationally expensive: the sensor needs to be

operated many times for each signal of the training data set. On the contrary, the readout layer can be trained with a much larger size of a training data set because the second phase is less computationally expensive: the sensor needs to be operated only once for each signal of the training data set. Additionally, training the readout layer with a much larger data set can improve the statistics of calculating the probabilities in the database. Secondly, training the sensing setup in two phases offers the flexibility to choose whether to implement each phase either theoretically or experimentally. For example, a drive signal can be learnt in the first phase theoretically by numerical simulations and the readout layer can be learnt by experiments in the second phase. Otherwise, both phases could be implemented theoretically or even both phases could be implemented experimentally.

In paper VII, two combinations have been demonstrated, a theoretical and an experimental demonstration. In the theoretical demonstration, both phases have been implemented theoretically. In the experimental demonstration, the first phase has been implemented theoretically and the second phase experimentally. The second phase is implemented experimentally by considering a pipette which has been developed in another work. [29] With this pipette, it is possible to provide pre-defined zinc concetrations in a sequence programmed with a control software. This pipette is used in paper VII for the experimental demonstration. For both demonstrations, two patterns of zinc variations are considered; **class 0:** stable environment with minor changes in the zinc concentration; **class 1:** varying environment with noticeable changes in zinc concentration.

Theoretical demonstration: In this demonstration, in the first phase 10 environment signals are considered per class. To solve the optimization of finding the best drive v_*, the space of periodic square-wave modulated drive signals was sampled with plateaus of length τ and period 5τ. Both the plateau values and the period were optimised. It is important here to notice that τ should be smaller than the relaxation time of the system which is around 20 seconds. If $\tau > 20$ seconds, then, the system would be very likely to relax to stationary conditions whatever form the zinc concentration looks like.

In the second phase, the training set consisted of a much larger number of environmental conditions ($M = 1000, N = 2$). In this step, a table with the occurrences of all $n_c = 15$ letter words has been built by counting how many times a specific word occurs when the system is simulated under the optimal drive v_* and all the signals of the training set. Those frequencies can be used to calculate the probability that the sensor is exposed to classes 0 and 1 when a specific word occurs.

The theoretical demonstration in paper VII shows that there are words that occur frequently, and when they do occur, they occur predominantly under a specific environment class. The words were clustered in four groups. The first group can be used to identify class 0 with great fidelity. In this first group, words have been found which are periodically repeated with a period of five characters. The second group can be used to identify class 1. In this group, words have been found where this periodicity is broken. In particular, within 15 characters, the periodicity is disturbed at least once. The third group illustrates the case where there is roughly an even distribution, it might be hard to tell, and the last fourth group contains words with too low frequency that signify nothing.

Remarkably, in paper VII, it is shown that a reservoir with only one sensor can be taught to “speak” sentences about the ionic environment with a sufficient amount of structure so that one can distinguish between the zinc variation patterns. As shown in this work, during some pulses, the signal $I[u, q](t)$ is weak and it is hard to discern whether it is going up or down, *i.e.* whether the character "u" or "d" should

be assigned. The system can be optimized to encode information about the environment in such weak signals. However, only the automated algorithm can discern such weak signals because it is hard to see by the naked eye.

The experimental demonstration: The first phase was implemented with the same procedure as in the theoretical demonstration, by numerical simulations, with some differences because the numerical simulations should mimic the response of the real device in the laboratory. This is significant for a synergy between first phase (theory) and second phase (experiment). In the second phase, two experiments were implemented per environmental conditions, four in total. For every experiment, the OT sensor was driven by the optimal drive found in the first phase. Experiments of the second phase were implemented by varying the zinc concentration around the OT sensor with the multifunctional pipette. [29]

Words with $n_c = 10$ characters have been identified and accompanied with the number of their occurrences under each class of environment. Based on their occurrences they have been divided into three groups. Group 1 consists of all the words that appear some times under class 0 and do not appear at all under class 1. The words of group 1 could be used as bar codes of class 0. Group 2 consists of words which could be used as bar codes of class 1. These words appear most of the times under class 1 but they do also appear a few times under class 0. In group 3, there are words that appear almost the same number of times under both classes. Those words could be hardly used as bar codes of either class.

However, it is worth noting that output currents under environment class 0 and under environment class 1 were found to have different scales. Therefore, there are some experimental conditions which were different for two different experiments. These differences in experimental conditions were not considered in the model used for the numerical simulations implemented in the first training phase. This finding makes it clear that to combine numerical simulations in the first training phase with experiments in the second training phase one needs to have an accurate model of all the experimental conditions.

Chapter 4

On reservoir computing with memristor networks

The findings from the previous section can be used to generalize, to further develop methods for temporal processing problems such as time series classification and prediction. So far, in the previous sections, a simple classification problem of sensing has been disucssed, *i.e.* to classify whether the environment signal is stable or varying. The methods are generic, and one can use the same methods outside of the sensing context. In fact, sometimes it is hard to draw a line between the two. Naturally, the signal q used to describe the environmental condition can be considered as the input to the classification problem. For example, electrocardiogram signals could be considered and then the task could be to classify if an electrocardiogram signal belongs to a healthy or diseased subject. This chapter describes how memristor networks can be used in the traditional machine learning sense.

The memristor's resistance, memristance, is bounded between a maximum and a minimum value. The fact that the memristance is bounded is a reason why memristor is a non-linear element. This is explained in section 4.1.

In section 4.2 a measure of reservoir computing capacity is presented. This measure has been developed and published in paper V. It can be used to compare different reservoirs based on their ability to occupy different regions of the phase space under different input signals, *i.e.* to separate inputs. It is tested on memristor networks and conclusions are drawn regarding strategies of designing optimal memristor networks in the reservoir computing context.

In section 4.3, generic methods are presented for improving the computing capacity of dynamical systems. Those methods have been developed in paper VIII and have been demonstrated with one memristor element on an electrocardiogram (ECG) signal classification task.

In section 4.4, it is shown that memristors can be used to build devices for predictions. In paper VI, a memristor has been trained for predicting at an early stage whether a patient has the sepsis disease. This method is extended in this thesis to consider networks of many memristors.

4.1 Memristor as a non-linear element

If one wishes to use memristor elements for sequence classification then it is important to assure that the memristance approaches the boundaries at least once as the reservoir evolves in time. This is shown by a hand-drawn example in Fig. 4.1: the sequence of voltage pulses across a memristor element can be encoded into the memresistance, provided the resistance has approached the boundaries during the application of the input signals.

In this example, a memristor model is exposed to two sequences of bits "1010" and "1100" (notice the same amount of 0s and 1s in both strings but with different order). Those bit sequences are converted into voltage signals V_1 and V_2 across the memristor model as it is shown in panels a) and b). In panel a), the parameter $\beta = \beta_1$ of the memristor is quite small and the resistance value after applying either of the input sequences is driven towards the same value $R_{1F} = R_{2F}$. Therefore, it is impossible to infer which of the two sequences was applied by only reading the resistance value R_{1F}. Since the parameter $\beta = \beta_1$ is quite small, the resistance does not approach the boundaries for either of the input sequences. Therefore, non-linearity is never exploited, which implies that with this choice of parameters the system is rather "dull" from the information processing of view. Non-linearity normally implies "inteligence".

In contrast to the above case, in panel b), it is shown that the sequence of applied voltage pulses matters when the memristor has approached its boundaries. In this example, the parameter $\beta = \beta_2$ is assumed much larger and the resistance value is not the same after applying either of the input sequences ($R_{1F} \neq R_{2F}$). Since parameter β is much larger the resistance approaches the boundaries under both sequences. Therefore, it is possible to encode the sequence into the resistance value: if one reads a resistance value closer to R_{1F} (R_{2F}) then would infer that the input sequence is "1010" ("1100").

4.2 Measuring the reservoir's computing capacity

In paper V, a measure has been developed which can be used to compare different reservoirs based on their computing capacity. The computing capacity of a reservoir is defined as the ability of the reservoir to separate any pair of input signals. It is assumed that for a pair of input signals i_1 and i_2, the system adopts the states $R[i_1](T)$ and $R[i_2](T)$ after being exposed to the input for a time T. It can be claimed that the larger the distance $||R[i_1](T) - R[i_2](T)||$ between those regions is, the better the reservoir separates the inputs i_1 and i_2. The distance $||R[i_1](T) - R[i_2](T)$ for each pair of inputs i_1 and i_2 is defined as:

$$||R[i_1](T) - R[i_2](T)|| = \left(\frac{1}{N_R} \sum_{m=1}^{m=N_R} (R_m[i_1](T) - R_m[i_2](T))^2 \right)^{(\frac{1}{2})} \tag{4.1}$$

By considering N_p input pairs, N_p distances can be calculated for every reservoir of interest. By doing statistics on those N_p distances it is possible to quantify the ability of a reservoir to separate inputs. In paper V, those distances are used to create a graph with the cumulative frequency of the distances. A hand-drawn example of such a graph is given in Fig. 4.2 where the cumulative frequency Φ_5 is shown for all the calculated distances σ_5. The minimum distance can be $\sigma_5 = 0$ and the maximum distance $\sigma_5 = R_{max} - R_{min}$. This graph shows that Φ_{5i}i100 % of distances are equal or smaller than σ_{5i}.

Another hand-drawn example is shown in Fig. 4.3. This example can be used to explain how the graph of the cumulative frequency Φ_5 can be used to compare different reservoirs. There, the graph of the cumulative frequency Φ_5 is shown for three different reservoirs and is generated by the same pairs of inputs. "Reservoir 1" performs worse than the other reservoirs: many pairs of inputs have not been separated at all with $\sigma_5 = 0$. "Reservoir 2" performs better than "Reservoir 1": A less number of input pairs has been separated with zero distance and most pairs

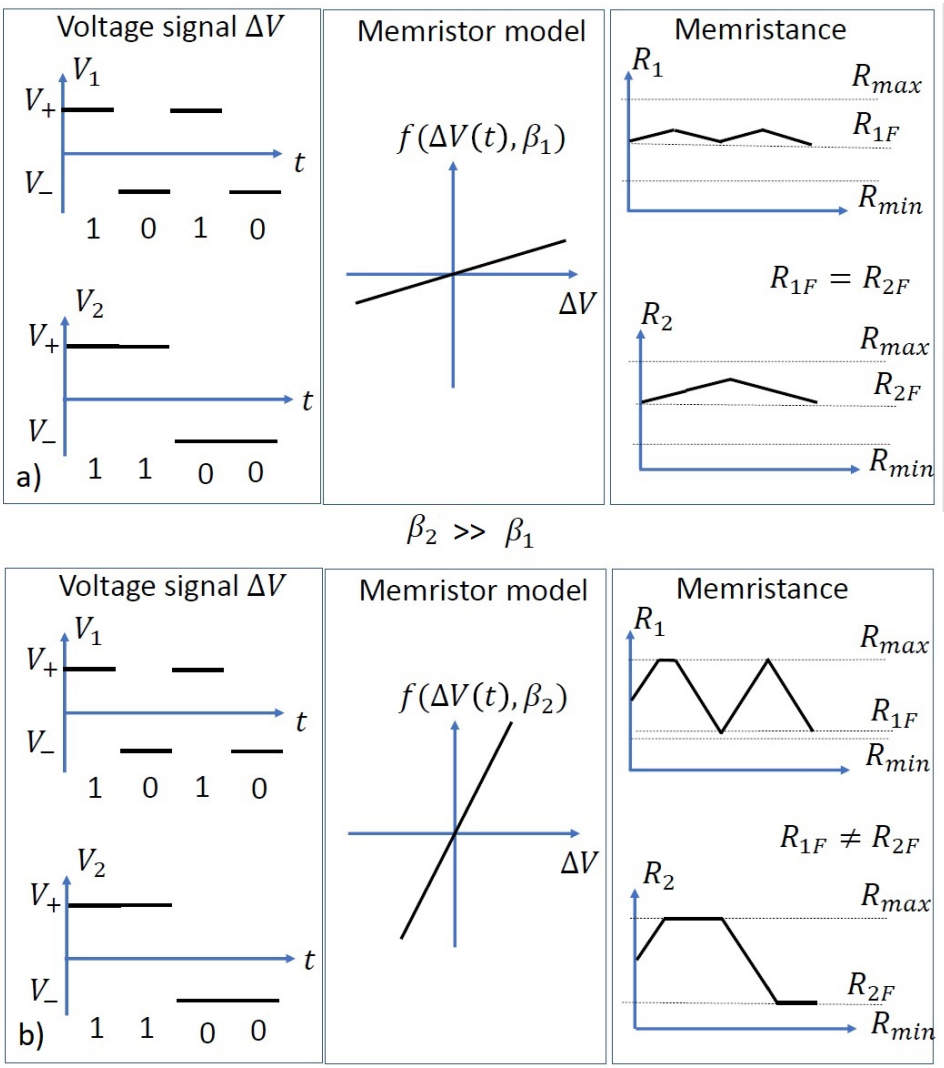


FIGURE 4.1: A hand-drawn example of the memristance behavior under two different cases and $V_{thr} = 0$. In panel a), the memristor model has a parameter $\beta = \beta_1$ and in panel b), $\beta = \beta_2 >> \beta_1$.

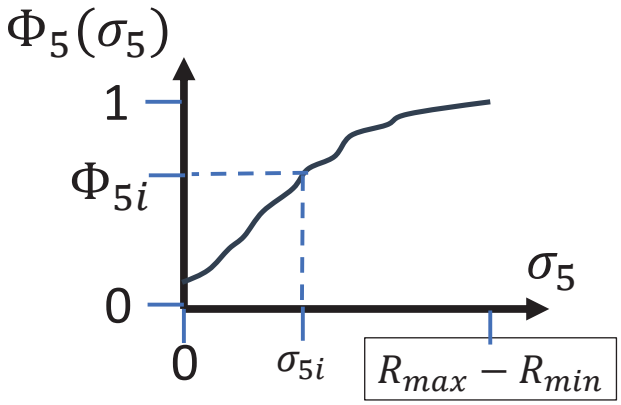


FIGURE 4.2: A hand-drawn example. On the vertical axis the cumulative frequency Φ_5 of all the distances σ_5 in the horizontal axis. It is shown that $\Phi_{5i} \cdot 100$ % of distances are equal or smaller than σ_{5i}.

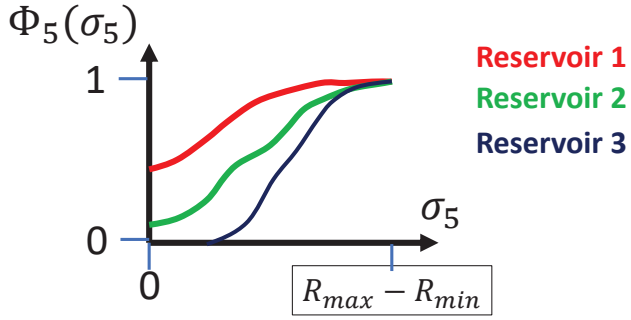


FIGURE 4.3: A hand-drawn example. The cumulative frequency Φ_5 with regards to distances σ_5 for three different reservoirs, Reservoir 1, Reservoir 2 and Reservoir 3.

have been separated with a larger distance. "Reservoir 3" performs better than all the other reservoirs: Not even one pair of inputs is separated with zero distance and most pairs are separated with larger distances than "Reservoir 1" and "Reservoir 2".

In paper V, the cumulative frequency has been used to compare six memristor networks on their ability to separate inputs. Six memristor networks have been considered which consist of non-identical memristors. Five of the networks have the same structure as the networks $N1$, $NSP2$, $NSP4$, $NSP6$ and $NSP8$ which have been considered in paper II. Those networks are called in paper V as "Network 1", "Network 2", "Network 3", "Network 4" and "Network 5" respectively. The sixth network is randomly generated with 16 memristors and random connections between them. This network is called as "Network 6".

The cumulative frequency of those networks for 1000 pairs of inputs is shown in Fig. 4.4, which is a figure taken from paper V. Panel (a) depicts a broader range of the distances. Panel (b) emphasizes the important region for distances close to zero. Thus of the two, panel (b) is the most important one. From the separability point of view, it seems that Network 5 performs best (the black full line is the lowest of all curves), followed by the second best, Network 4. Interestingly, Networks 6 and 3 are very close in their power to separate inputs, though they consist of vastly different number of memristors. This shows that the size of memristors does not necessarily matter for this choice of networks. Network 6 is the largest of all, but still does not offer the best performance. The best performing Network 5 is not as big as Network 6 but easily outperforms it.

Network 5 and Network 6 have different structures and that might be the main reason of their different power on separating inputs. Network 5 is constructed in a way that all the memristors are exposed to a variety of voltage differences. Each memristor in the network is roughly exposed to a different voltage range of the drive $u(t)$ depending on its position in the network. Why is this so? The memristor network is essentially a voltage divider.

Due to the structure of Network 6, almost all the memristors are likely to be exposed to similar scales of the input voltage difference, and further typical voltage differences they experience are rather small. Therefore, almost all the memristors in Network 6 are expected to have a relatively small response to the input signal and not to separate inputs at a large extent.

The results in paper V agree with the results obtained in paper II regarding the computational power of Network 2, Network 3, Network 4 and Network 5 (which have the same structure as $NSP2$, $NSP4$, $NSP6$ and $NSP8$ respectively of paper

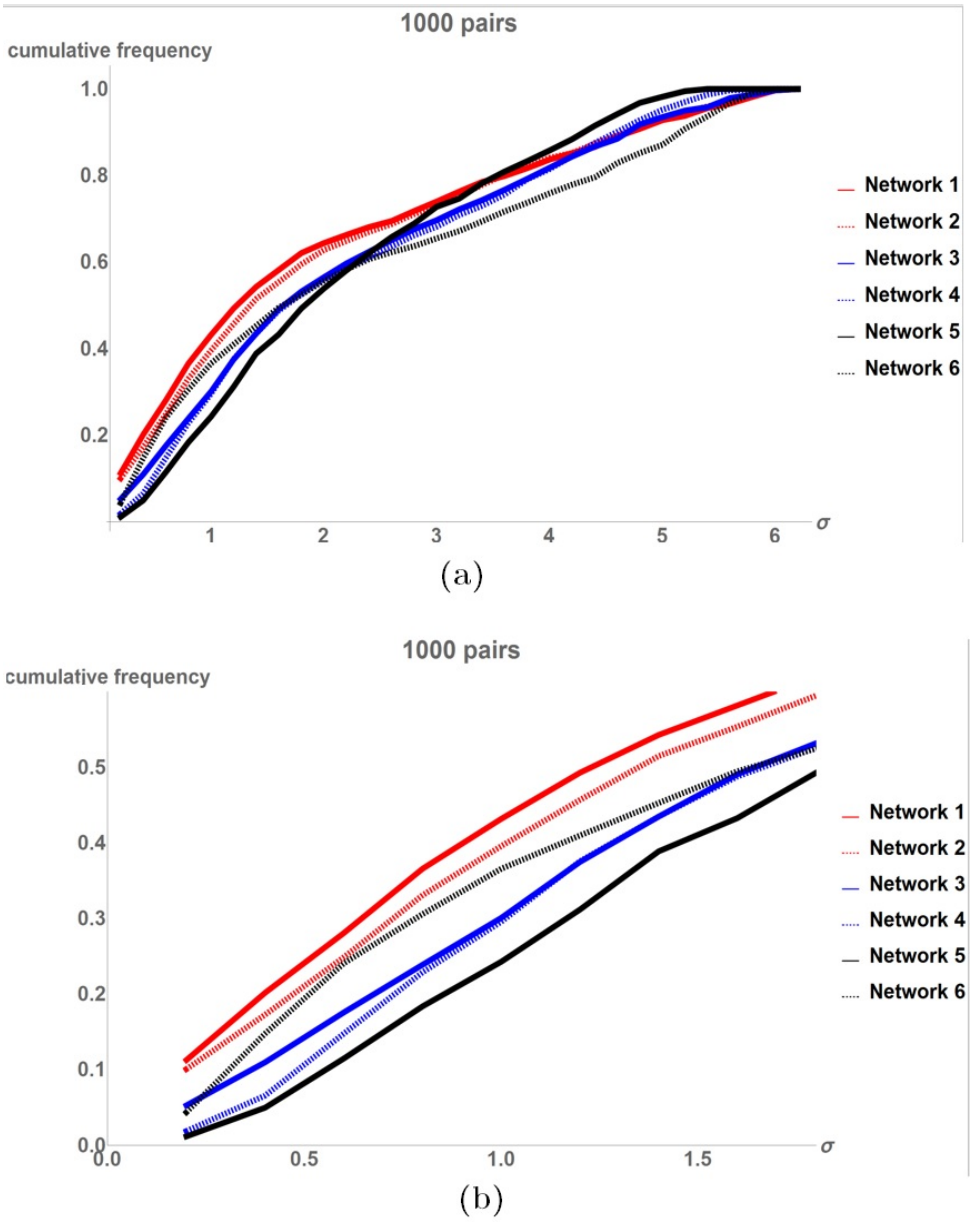


FIGURE 4.4: Figure taken from paper V. The cumulative frequency of the six considered networks in a) the whole range of distances and b) in the range of low distances.

II). In paper II, the largest separability index on the training data was found for $NSP8$. Similarly, in paper V, network 5 was found with the best power to separate inputs. Similar conclusions are drawn for the other networks. The network $NSP6$ fitted better on the training data than $NSP4$ in paper II. Similarly, Network 4 was found to separate better inputs than Network 3. Therefore, both papers agree that input separation is favored by adding memristors in series and parallel, direction 3 in paper II.

However, there are some drawbacks regarding direction 3 in paper II. To begin with, the algorithmic complexity of simulating electronic circuits is worse than linear with regards to the number of nodes in the circuit. Therefore, by increasing the number of elements in this direction, the execution time of numerical simulations is not linear in the number of elements. It is supra-linear in the number of elements. However, this only holds for software models. This would be of no concern if memristor networks were operated as pieces of hardware. Additionally, by increasing further the number of elements and considering networks with 10, 12, $\cdots$ memristors, some memristors would be exposed to a limited range of typically small voltage differences and would not be responsive at all (the voltage divider property). The only way to increase the number of memristor elements without memristors being exposed to smaller voltage scales is to add new memristors in a circuit by using parallel connections. Increasing the number of memristors in such a way is studied in the section 4.4 of this thesis. However, an infinite number of memristors connected in parallel is not feasible either in hardware implementations because circuits with many resistances in parallel would require powerful voltage sources that could pump very large currents through the circuit. Therefore, one should try to find memristor network structures with maximum input separation but with the minimum amount of memristor elements.

4.3 Improving the computing capacity of dynamical systems

Instead of modifying the internal structure of a dynamical system one can try to adjust the input that is being fed into the system to achieve optimal separation. In paper VIII, the findings of the previous papers I and II are further developed for considering reservoirs with the minimum number of elements. Typical of reservoir computing is to implement a neuromorphic computation by considering a large amount of interconnected elements. However, in paper VIII, it has been shown that by using a reservoir with only one memristor, and an optimised drive signal, more than 93% of electrocardiogram signals can be classified correctly.

Two implementations of the ideas discussed in this thesis are shown in paper VIII. Firstly, in **Implementation 1** the reservoir is stimulated by a drive signal u and an input signal q. The dynamics of the reservoir is defined by $\dot{R}(t) = H(R(t), q(t), u(t))$. Secondly, in **Implementation 2**, the complexity of the system can be increased by adding a feedback, $\dot{R}(t) = H(R(t), q(t), u(t) + u_R(t))$, where the feedback signal $u_R(t)$ is a function of the reservoir state $u_R(t) = h(R(t))$. Note that H is considered fixed, and only once H is given, one chooses the appropriate implementation.

To find an optimal drive signal u, the same training procedure as the one used in paper VII is considered, a genetic algorithm optimisation. The system is trained on a set of data. The ability of the system to generalise is examined by using a separate set of test data.

In paper VIII, the training procedure consists of two phases. Firstly, the drive signal and the feedback (if used) are trained without considering the readout layer.

In this phase, the goal is to find the optimal drive signal and the feedback function (if used) that achieves the maximal trajectory separability. The separability index ν has been used to measure trajectory separability. Secondly, only the readout layer is optimised, by keeping the drive signal and the feedback function found from the first phase.

Herein, it is important to notice that the readout layer implementation used in paper VIII differs somewhat from the central reservoir computing dogma. In reservoir computing, the readout layer should be memory-less. However, in paper VIII, the readout layer averages the memristance values over a time interval. Inference is done by a linear combination of the average memristance values instead. The average memristance values are provided to the readout layer because the separability index ν accounts for the average memristance values. Therefore, maximising the separability index ν indicates the maximisation between the average values of memristances. When compared to the classical reservoir computing readout layer, the computational cost of implementing such a unit is marginal.

Two options of how the input signal $q(t)$ interacts with the reservoir are considered. In **option 1**, the input signal influences the β parameter of the memristor model similarly to the assumptions made in papers I and II when memristor networks were considered for sensing. In **option 2**, it is assumed that the input signal acts as an external voltage source. In this option, the parameter β of the memristor is kept fixed. These two options (Options 1 and 2) along with the two implementations (Implementations 1 and 2) result in the four models I1O1, I2O1, I1O2 and I2O2.

The performance of those four models has been compared with a set of simpler models in which neither a drive signal nor a feedback function are optimised. Those simpler models reveal a "raw" intelligence of just a memristor reservoir. Those models are used to answer the question: what would be the computing capacity of a single memristor reservoir without training any drive signal or feedback function.

The models have been trained and tested on a labelled data set of electrocardiogram (ECG) signals. A binary classification problem has been investigated. The strategy of using the single memristor for ECG signal classification is simple to describe: the memristance ought to be driven towards R_{min} or R_{max} depending on whether the input signal belongs to either of the two classes. If this can be achieved, classification can be performed by simply checking whether the average memristance value exceeds a pre-defined threshold.

In the training procedure, 40 signals from each class have been used. The ability of the trained models to generalise has been validated on a separate set of test data where 740 signals have been used from each class. The quality of recognition is described in terms of the success rate being defined by the percentage of correct signal classifications of the test data set. In paper VIII, the models have been tested on the worst case when few training examples are available. The set of test data has been considered to be much larger than the set of training data ($740 >> 40$). It has been shown that reservoir computing performs relatively well with a few training examples [30, 31] and this is also investigated in paper VIII. Reservoir computing can extract general features from the training data set because a small number of parameters needs to be learnt.

The downloaded ECG signals, which are used in paper VIII, are all synchronized according to the QRS peak. The top of the QRS peak is a natural time reference for all ECG signals. The models are also trained with non-synchronised signals. This is a harder since the phase of ECG is unknown to the trained models. For this purpose, the signals have been randomly shifted in time so that they are all non-synchronised

to each other. Thus the models are tested on two different major cases, when the ECG signals are aligned and when the signals are asynchronous.

Additionally, all the ECG signals have been converted with linear mappings into signals of the memristor's β parameter in the case of **option 1** and into voltage source signals in the case of **option 2**. Those linear mapping are called in this thesis as the input layer. One could argue that training the input layer could be important for improving the performance of the models. This argument can be based on the hand-drawn example of the section 4.1 which shows that it is important that the memristance approaches the boundaries during the application of input signals. Modifying the input layer would result in a modified way that the memristance approaches the boundaries and therefore a different non-linear functionality of the memristor element. In paper VIII, the additional computing capacity by training the input layer is shown in the results section. Firstly, the input layer has been kept fixed. Secondly, the input layer has been trained additionally to the drive signal and the feedback function.

4.3.1 Fixed input layer

The maximum separability index in the first phase of the training procedure and the percentage of correct classifications is shown in Table 2 of paper VIII for both aligned and asynchronous ECG signals.

For the aligned signals, it was found that:

- A larger separability index ν was obtained for the models with a feedback mechanism (I2O1 and I2O2) than the ones without feedback (I1O1 and I1O2). The lowest separability indexes were obtained for the models I0O1 and I0O2 where neither a drive nor a feedback were optimised.

- All the success rates with optimised input features (I1O1, I2O1, I1O2 and I2O2) were found $S > 93.2\%$ indicating that these or similar models can be used for classifying aligned ECG signals.

- The success rates were smaller for the models with an additionally optimised feedback than the models with just an optimised drive signal. This indicates over-fitting. Training of an additional parameter (feedback) improved training procedure but failed to improve testing. The additional parameter was too powerful so that models fit too much on details of the training data which are not generally met on the test data.

- The success rates for models with just a drive signal (I1O1 and I1O2) were found larger than the models without any drive signal (I0O1 and I002). In particular, by training a drive signal, the success rates improved from 87.2% to 97.9% and from 51.9% to 97.8%. Therefore, training a drive signal resulted to exceptional success rates which could not be achieved by just using one memristor element.

- Additionally, for the model I1O2, even though the training process resulted in a very low separability index, the success rate was large (97.8%). This happened because it was still possible to construct a readout layer which classifies most of the input signals correctly. By inspecting closer the memristance over time, we saw that although the instantaneous memristance $R(t)$ was driven to overlapping regions, the average value of the state was separable. This is one of the advantages of using a readout layer which averages the state over time.

For the asynchronous signals it was found that:

- The separability indexes ν and the success rates were found smaller for each model than when considering aligned signals. This was expected since now the models should be smarter and infer correctly independently of the input signal phase.
- Including and optimising a feedback mechanism (models I2O1 and I2O2) improved both the training procedure (separability index) and the success rate on the test data. Therefore, there was no over-fitting by using the asynchronous signals.

4.3.2 Optimised input layer

The maximum separability index in the first phase of the training procedure and the percentage of correct classifications is shown in table 3 of paper VIII.

The comparison between the results of a fixed and an optimised input layer has shown that:

- The best success rates were found when the input layer was optimised. For the aligned signals, it was found 98.6% and for the asynchronous signals 93.1%.
- Over-fitting was noticed since the additional training of the input layer resulted in larger separability indices but smaller success rates.
- By only training an input layer resulted in success rates smaller than 90%. The only way to achieve $S > 90\%$ was to provide an additional input feature such as a drive signal or a drive signal with a feedback function.

4.4 Prediction models with memristor networks

In paper VI, a setup with one memristor element has been suggested for predicting, at an early stage, whether a patient in intensive care unit (ICU) has the sepsis or not. For this purpose, data sets of clinical variables from ICU patients have been downloaded from the 2019 Physionet Computing in Cardiology Challenge (https://physionet.org/content/challenge-2019/1.0.0/). [32] The data consist of clinical variable values from 45643 ICU patients. An example of the data for each patient is shown in Fig. 4.5. There are 41 columns. From the 1st until the 39th column the values of clinical variables are given for each hour of staying in ICU. These hours are symbolised as X in Fig. 4.5. In the 40th column, the length of stay in ICU (hours) is given and in the 41st column the label of the data is given, a boolean value which is 1 if the patient was clinically diagnosed with sepsis six hours afterwards and 0 otherwise.

The data have been preprocessed with a standard score normalization. The mean value μ_j and standard deviation σ_j of each clinical variable have been calculated. Then, each clinical variable x_j is modified according to:

$$\frac{x_j - \mu_j}{\sigma_j} \tag{4.2}$$

In paper VI, an inference unit is used as a basic component. The setup of the inference unit is shown in Fig. 4.6 which is taken from paper VI. In this figure, the graphs are hand-drawn examples to illustrate the main ideas. The symbols $x_j(t)$

Index:	1	2	3	⋯	39	40	41
Medical parameter:	HR	O2Sat	Temp	⋯	HAT	ICULOS	Label
Values of medical parameters:	X	X	X	⋯	X	1	X
	X	X	X	⋯	X	2	X
	X	X	X	⋯	X	3	X
	X	X	X	⋯	⋮	⋮	⋮
	X	X	X	⋯	X	N_{los}	X

FIGURE 4.5: Data for a patient in ICU. In the first row, the indices of all the medical parameters are provided. In the second row, the names of medical parameters are provided, where HR stands for heart rate, O2sat for blood oxygen saturation levels, Temp for temperature, HAT for hours between hospital admittance and ICU admittance, ICULOS for length of stay in ICU and label is a boolean value with 1 if the patient was clinically diagnosed with sepsis six hours afterwards and 0 otherwise.

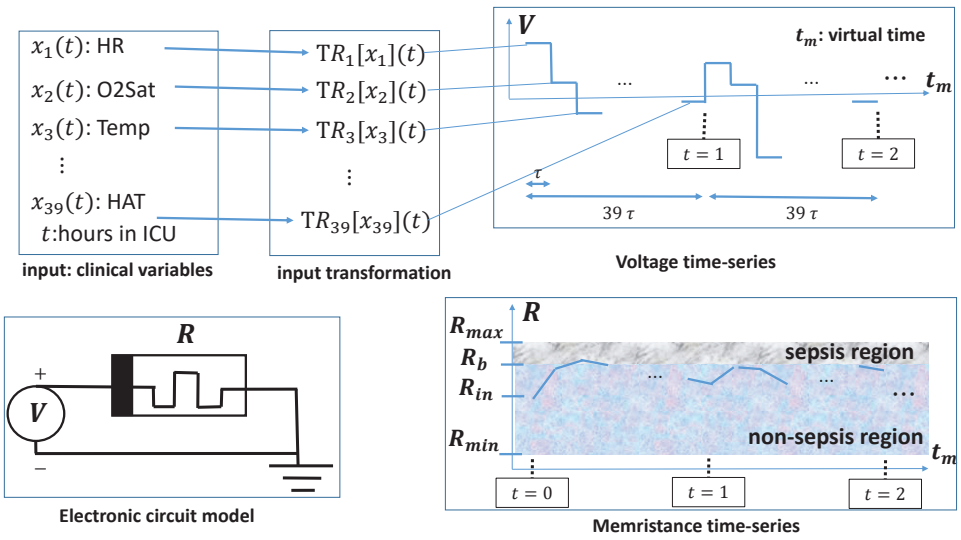


FIGURE 4.6: Figure taken from paper VI. The setup of predicting early with one memristor whether a patient has the sepsis.

with $j = 1, \cdots 39$ denote the 39 measured clinical variables. Data retrieved from 20643 (25000) patients have been used as a training (validation) data set.

The voltage signal across the memristor element is generated by transforming the clinical variables. A different transformation function $TR_j[x_j]$ has been used for each input x_j to produce a voltage pulse with a certain duration. By this way, 39 pulses would be created every hour of stay in ICU (as it is shown in Fig. 4.6). If the clinical variable $x_j(t)$ is not available at time t (NaN), then the transformation function returns zero *i.e* a decision cannot be made whether the memristance value should move towards the sepsis region or not. An example of transformation type could be the following:

$$\begin{aligned} TR_j^k[x_j](t) = w_{jb} + w_{j0}\, x_j(t) + w_{j1}\, x_j(t-1) + \cdots \\ + w_{jk}\, x_j(t-k) \end{aligned} \tag{4.3}$$

With such a transformation function, a linear correlation between chronically previous values of the same input can be accounted since $x_j(t)$, $x_j(t-dt)$, $\cdots$ are included. It is obvious that this correlation would include first, second, third *etc.* derivatives of the input x_j. The question is if non-linear correlations between the inputs $x_1(t)$, $x_2(t)$, $\cdots$ $x_{39}(t)$ are accounted. These correlations are possible to be accounted if the voltage signal is produced as many pulses per hour because the memristor is a non-linear element.

However, such a non-linear correlation would not be accounted if the voltage signal was produced as one pulse per hour of stay in ICU. An example of such transformation for producing one pulse $V_{pul}(t)$ per hour t would be:

$$V_{pul}(t) = w_0 + w_1\, x_1(t) + w_2\, x_2(t) + \cdots w_{39}\, x_{39}(t) \tag{4.4}$$

It is obvious that the above transformation function would account for only linear correlations between the inputs $x_1(t)$, $x_2(t)$, $\cdots$, $x_{39}(t)$ of the same hour t.

The produced voltage signal $V(t)$ is applied across a memristor element. The simulation of the memristance results in a memristance time-series $R(t)$. The memristance is always initialized at the value R_{in} at time $t = 0$. Then, the memristance changes depending on the applied voltage signal. The purpose of this setup is to encode the early diagnosis of sepsis in the memristance: For the specific example of Fig. 4.6, if the memristance is larger than the value R_b, then the unit predicts sepsis. Initially, when $R = R_{in}$ it is assumed that the probability of a patient to have the sepsis is 0 since the model has not been exposed to any input yet. Therefore, the sepsis region cannot involve R_{in}. Instead, the sepsis region should involve R_{max} or R_{min}. If the sepsis region involves R_{max} (R_{min}) then $R_b > R_{in}$ ($R_b < R_{in}$).

The goal is to train the transformation functions used and the parameters R_b of the models involved. To grade the fitness of a choice of such parameters, a utility score [32] has been used. This score is 0.0 if a system predicts always non-sepsis and 1.0 if the system predicts correctly all the cases of sepsis and non-sepsis. This utility score is additionally explained in paper VI and further details of calculating this score can be found in [32].

4.4.1 Predicting with one memristor

Firstly, one inference unit was trained with a procedure explained in paper VI. In this procedure, a genetic algorithm optimization has been used by using both the training and validation data sets. The goal has been to train the transformation functions

and the parameter R_b. By using transformation functions TR_j^0, the best utility scores during the training procedure were found as 0.3140 (0.2411) for the training (validation) data. The utility score 0.3140 corresponds to around 80% correct predictions of non-sepsis labels and 58% correct predictions of sepsis labels.

Many inference units $I_1, I_2, \cdots, I_N$, different to each other, can be also combined to improve the prediction accuracy. In paper VI, inference units have been used in parallel to improve the prediction accuracy. If each inference unit I_i is trusted with a probability tr_i, where $\sum_{i=1}^{N} tr_i = 1$ and predicts sepsis with a probability Pr_i, then, the whole system would predict sepsis with a probability:

$$Pr^N(R) = \sum_{i=1}^{N} Tr_i \; Pr_i(R) \tag{4.5}$$

In paper VI, an algorithm has been suggested for training sequentially the inference units $I_1, I_2, \cdots, I_N$. In paper VI, up to three inference units were trained sequentially and the best utility scores were obtained 0.3241 (0.2501) for the training (validation) data set.

In paper VI, it has been shown that by additionally training a second and third model improved the utility scores on both training and validation data sets. A question is whether training additional models always improves the utility scores. In particular, would the utility scores always increase given that all the models I_3, I_4, $\cdots$ I_N consisted of the same transformation function? Moreover, which transformation function would be suitable? Another question is if the usage of more complex transformation functions, such as neural networks, could improve the utility scores. All these questions are to be answered in future work.

The best algorithm of paper VI was submitted on the 2019 Physionet Computing in Cardiology Challenge (https://physionet.org/content/challenge-2019/1.0.0/) [32] and received a utility score 0.200 on an unknown separate data set. This utility score corresponds to around 80% correct predictions of non-sepsis labels and 44% of correct predictions of sepsis labels. It is remarkable that this result has been obtained by just training a few hundred parameters. However, the best submission in this challenge obtained a utility score 0.364 which is far larger than 0.200.

Therefore, the suggested solutions of paper VI under-fit the training data set. Under-fitting the data set means that the suggested models are not complex enough to model non-linear details of the data. The question here is how should one use many memristors to model even better details of the data, *i.e.* to over-fit the data. To over-fit the data, a reservoir computing paradigm is suggested with memristors connected in parallel.

4.4.2 Predicting with many memristors

This section shows that networks consisting of non identical memristors connected in parallel [14] can be used for over-fitting the training data and achieve utility scores comparable to the largest ones obtained in the 2019 Physionet Computing in Cardiology Challenge.

The key idea is to exploit heterogeneity. Heterogeneous transformation functions may be used to develop different voltage signals. Then, the heterogeneous voltage signals may be applied across heterogeneous memristor elements. If the memristor elements operate by approaching their boundaries then each memristor introduces a non-linearity necessary for "intelligence". We ensured this happenen, in two ways:

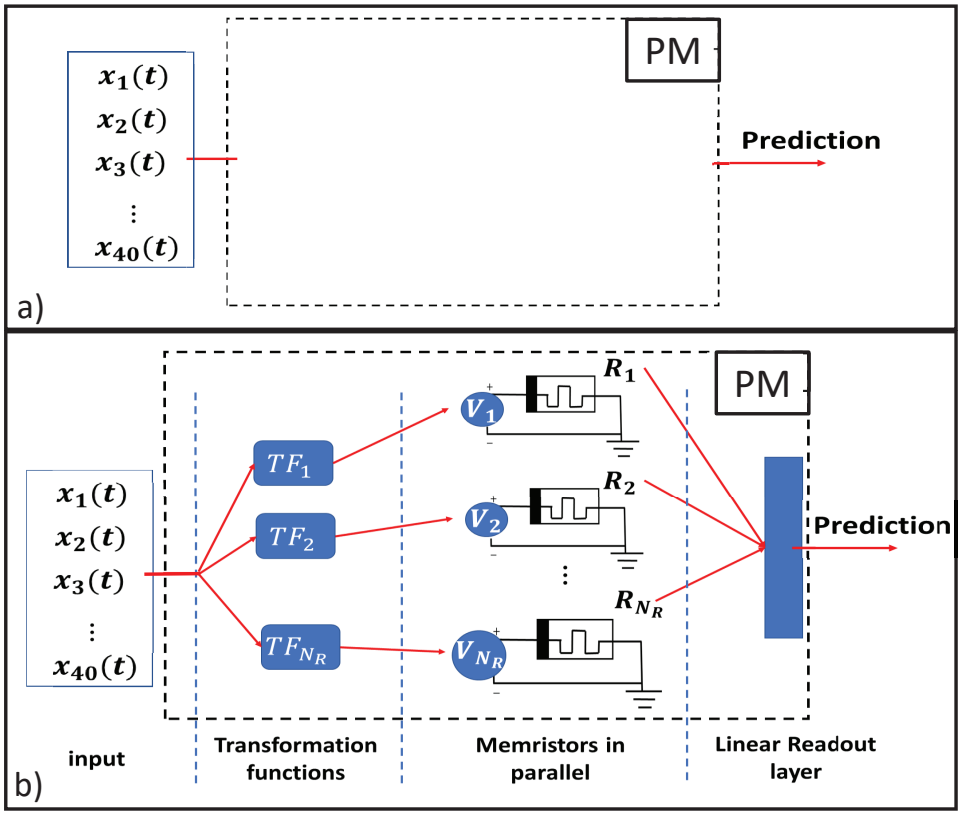


FIGURE 4.7: The prediction setup by using memristors connected in parallel. a) A PM is defined as a system with a vector input and output a prediction. b) The structure of the PM consists of transformation functions TF_i which produce voltage signals V_i, memristor models R_i and a linear readout layer which outputs the prediction at time t.

Firstly, a different voltage signal has been generated across each memristor. Secondly, the memristors are heterogeneous and hence the boundaries are approached in many different ways for each memristor. Finally, by only training a linear readout layer it is possible to extract important information from the reservoir for accurate predictions.

The prediction setup with memristors connected in parallel is shown in Fig. 4.7. In panel a), the generic setup is shown where a PM is herein defined as a model which receives as input at hour t the history of clinical variables $x_1(0)$, $x_2(0)$, $\cdots$, $x_{40}(0)$, $x_1(1)$, $x_2(1)$, $\cdots$, $x_{40}(1)$, $\cdots$, $x_1(t)$, $x_2(t)$, $\cdots$, $x_{40}(t)$ and produces a prediction as whether the patient is going to have the sepsis within the next 6 hours.

Note that data have been exploited differently in this section than in the previous one. In this section the fortieth clinical variable, length of stay in ICU, has been also used as input. In paper VI, this variable was assumed to be incorporated in the internal dynamics of the system. However, during the attendance of the conference "Computing in Cardiology 2019", it was pointed out by many attendees that length of stay in ICU is an important parameter and needs to be used as input. Additionally, the missing values "NaN" have been treated differently in this section of the thesis. Forward filling has been used to fill the missing values. If a clinical variable value is missing at hour t then the latest existing value is used, *e.g.* at time $t-1$ or at time $t-2$ *etc.*

In Fig. 4.7 b), one can see details of how the PM is constructed. At time t, the

input vector is transformed into the voltage signals V_1, V_2, $\cdots$ V_{N_R} with the transforming functions TF_1, TF_2, ..., TF_{N_R} respectively. In this section, a transforming function TF_i converts an input vector $x_1(t)$, $x_2(t)$, $\cdots$, $x_{N_R}(t)$ into a voltage signal V_i with sequence of pulses $w_{1i}\, x_1(t) + b_{1i}$, $w_{2i}\, x_2(t) + b_{2i}$, $\cdots$, $w_{40i}\, x_{40}(t) + b_{40i}$. The parameters w and b are randomly generated for each transforming function with values between -1.0 and $+1.0$.

The memristors are randomly generated too by a random choice of the parameters that describe the memristor model: V_{thr}, α, β and initial memristance value R_{in}. Here, it is important to notice that relatively large values of the parameters α and β have been generated to assure that memristors approach their boundaries. During each hour t, the average memristances of each memristor $\bar{R}_1(t)$, $\bar{R}_2(t)$, $\cdots$, $\bar{R}_{N_R}(t)$ are provided into the linear readout layer. The readout layer is used to calculate the following quantity:

$$lc(t) = w_0^L + \sum_{m=1}^{m=N_R} w_m^L\, \bar{R}_m(t) \tag{4.6}$$

If $lc(t) > 0.5$, then the prediction is sepsis, otherwise, the prediction is non sepsis.

Forward propagation of the PM: In this paragraph, it is explained how the PM is operated when data from a patient with index ind is imported as input: $x_1^{ind}(t)$, $x_2^{ind}(t)$, $\cdots$, $x_{40}^{ind}(t)$ for $t = 0, 1, 2, \cdots$. The corresponding label of this input vector is denoted as $x_{41}^{ind}(t)$ and has value either 0 or 1. Before, the operation, the memristances are initiated at the initial value of each memristor R_{in}. Initially, the PM is operated at $t = 0$. The voltage signals across the memristors are produced through the transformation functions and are applied. This results in updating the memristance and recording the mean memristance values during $t = 0$, $\bar{R}_1^{ind}(0)$, $\bar{R}_2^{ind}(0)$, $\cdots$, $\bar{R}_{N_R}^{ind}(0)$. Afterwards, the PM is operated at $t = 1$, the voltage signals for $t = 1$ across the memristors are produced through the transformations functions and are applied. The memristances are updated and the new values are recorded: $\bar{R}_1^{ind}(1)$, $\bar{R}_2^{ind}(1)$, $\cdots$, $\bar{R}_{N_R}^{ind}(1)$. This procedure is repeated at every hour t and all the values $\bar{R}_1^{ind}(t)$, $\bar{R}_2^{ind}(t)$, $\cdots$, $\bar{R}_{N_R}^{ind}(t)$ are recorded. For each hour, the linear readout layer predicts whether the patient has the sepsis by calculating the quantity:

$$lc(t, ind) = w_0^L + \sum_{m=1}^{m=N_R} w_m^L\, \bar{R}_m^{ind}(t) \tag{4.7}$$

where if $lc(t, ind) > 0.5$, then the prediction is sepsis, otherwise, the prediction is non sepsis.

Training procedure: The parameters w_0^L, w_1^L, w_2^L, $\cdots$, $w_{N_R}^L$ are the only trainable ones of the PM. The PM is operated only once under each available input and all the memristance values $\bar{R}_m^{ind}$ are summarised in a state space matrix as it is shown in Fig. 4.8. The multiplication between the state space matrix and the readout layer weights vector should be approximately equal to the labels. For this approximation to occur, the weights are found by least square error regression.

A problem would occur with least square error regression if the number of 0 labels is different than 1 labels. In particular, in the provided data sets for sepsis prediction, the number of 0 labels is quite larger than 1 labels. Therefore, solving the least square error regression would heavily approximate 0 labels than 1 labels.

In this thesis, to approximate with equal importance both 0 and 1 labels, the elements of the state space and label matrix are modified when the corresponding label is 1. Assuming that the number of 0 labels is denoted with nr_0 and the number

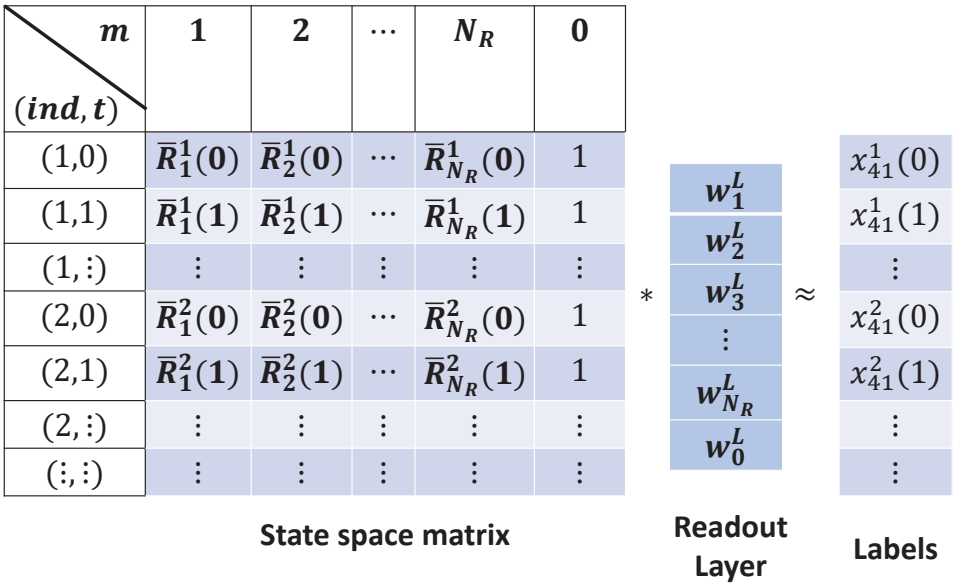


FIGURE 4.8: The state space matrix consists of all the average memristances $\bar{R}_m^{ind}(t)$ of the memristor m, patient ind and at time t. The readout layer is a linear readout layer with a vector of all the trainable parameters w_m^L. The state space matrix multiplied with the readout layer vector should approximate the labels $x_m^i nd(t)$

of 1 labels is denoted with nr_1, then the state space and labels matrix are modified as follows.

- For each ind and t
  - If label $x_{41}^{ind}(t) == 1$,
    - For $m = 1$ to N_R
      - Modify $\bar{R}_m^{ind}(t) := (\frac{nr_0}{nr_1})^{\frac{1}{2}}\ \bar{R}_m^{ind}(t)$
      - Modify $x_{41}^{ind}(t) := (\frac{nr_0}{nr_1})^{\frac{1}{2}}$
    - End-For
    - Replace element at row (ind, t) and column 0 with $(\frac{nr_0}{nr_1})^{\frac{1}{2}}$
  - End-If
- End-For

Then, the weights of the readout layer are calculated with least square error regression by using the modified state space and labels matrices instead.

Training PMs: Many PMs have been trained with variable number of memristor elements. The training procedure has been implemented for PMs with number of memristors 40, 80, 120, 180, 260, 350 and 500. The training data set obtained from 20643 has been used.

The utility scores after performing the training procedure on each PM are shown in Fig. 4.9. When the number of memristors per PM increases, then better utility scores are obtained. A utility score 0.353 was obtained with 120 memristors per PM. This utility score is larger than the ones obtained with one inference unit in paper VI, $\approx$ 0.314. Therefore, a PM with 120 randomly generated transforming function and memristors performed better than one inference unit which consists of one optimised transforming function and one memristor element. Additionally, with 500

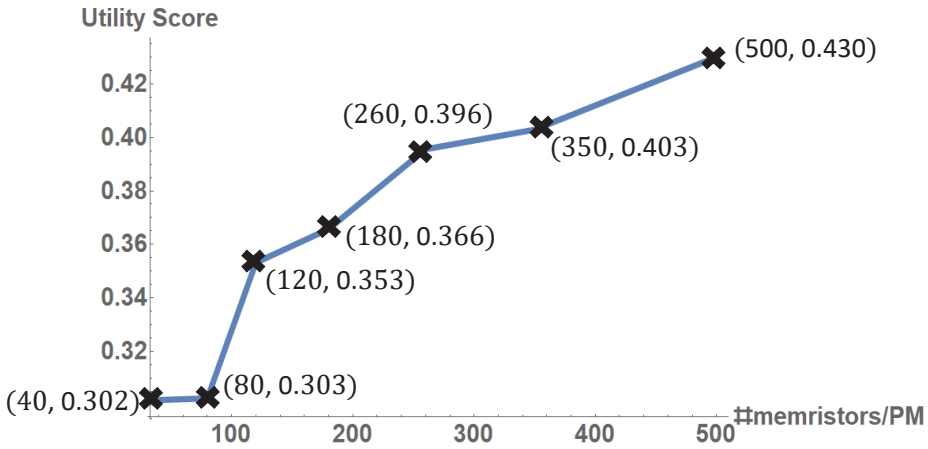


FIGURE 4.9: The utility scores after performing the training procedure. The utility scores are shown on the vertical axes and the number of memristors per PM is shown in the horizontal axes.

memristors per PM, a utility score 0.43 was obtained which is comparable to the best utility score obtained in the 2019 Physionet Computing in Cardiology Challenge. The utility score 0.43 corresponds to 77.4% correct predictions of the hours when there was no sepsis label (true negatives) and 72.1% correct predictions of the hours when there was sepsis label (true positives). It is remarkable that this was achieved by training only 501 parameters.

The results in Fig. 4.9 show that training PMs with even larger number of memristors can contribute to an even better fitting to the training data set. If one wants to fit even better the training data set, then, PMs with larger than 500 memristors could be considered. However, one limitation noticed during the numerical simulations was that the algorithmic complexity of the least square regression was supralinear in the number of memristor elements. In particular, training the PM with 80 memristors costed much more than double execution time when training the PM with 40 memristors. Therefore, if one wants to continue in the direction of training PMs with much larger number of memristors then faster algorithms are required for solving the regression problem, such as gradient descent optimisations. However, least square regression always guarantees on finding the best solution while gradient descent optimisation algorithms do not guarantee that [33].

A boosting algorithm: Fitting to the training data set can be further improved if a boosting algorithm is used. A second PM can be trained to predict correctly what the first PM has failed to predict. Then, a third PM can be trained to predict correctly what the other two PMs have failed to predict together. This process can be continued further.

In this thesis, a system consisting of many PMs is suggested as it is shown in Fig. 4.10. The input vector is supplied to each PM. Each PM updates its internal state, which are the memristances of their memristors, and returns as output a prediction as 0 (non sepsis) or 1 (sepsis). Another unit, called "Voting" in the figure, reads all the predictions and returns an overall prediction. If the majority of PMs' prediction is 1 (sepsis), then "Voting" returns 1 (sepsis), otherwise it returns 0 (non sepsis). If the number of 0 predictions is equal to the number of 1 predictions then the overall prediction is 0.

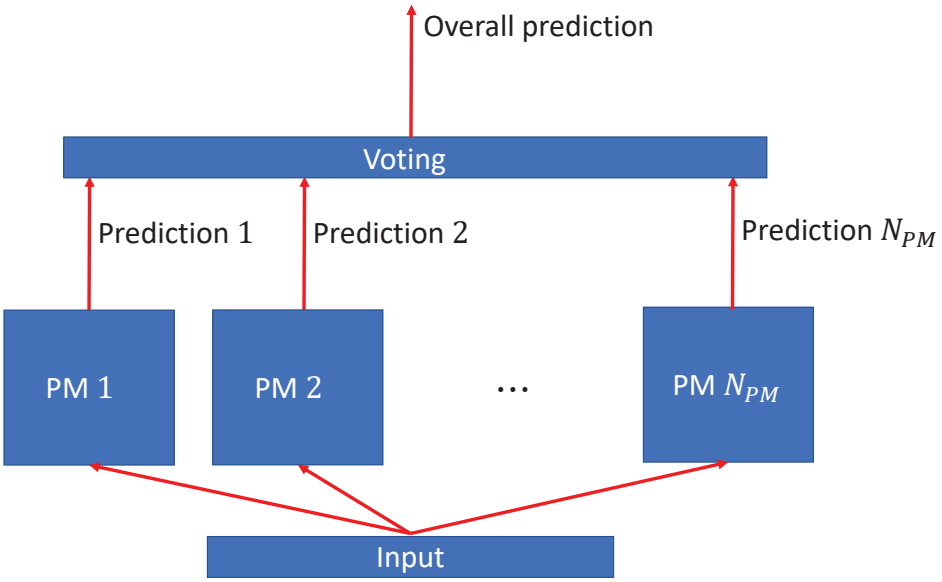


FIGURE 4.10: A system consisting of many PMs.

The training procedure: The PMs are trained sequentially. Firstly, PM 1 is trained as it has been explained before for the training procedure of one PM. Then each PM is trained sequentially to correct wrong predictions from the majority of the previously trained PM. For each PM, training is implemented with least square error regression but by accounting only the rows of the state space matrix that the overall prediction can be corrected by training one additional PM. To clarify the latest sentence, the following examples are given:

- First example: label 0 and predictions of the previously trained PMs 0, 1, 0, 1. This case is of interest. It is important for PM 5 to output prediction 0 because if it predicts 1 then the overall prediction would be wrong 1.
- Second example: label 0 and predictions of the previously trained PMs 0, 1, 1, 1. In this case, even if PM 5 predicts 0, then, the overall prediction would be 1. This case would be considered of no interest because a correction of the overall prediction is not possible. The key idea behind this strategy is that the least square regression error should focus on maintaining other correct predictions or correct others that can be corrected by only one additional PM.
- Third example: label 0 and predictions of the previously trained PMs 0, 0, 0, 1. This case would be also considered of no interest. Even if PM 5 predicts wrong 1, then the overall prediction would still be correct 0.

This training procedure has been implemented for a sequence of 20 PMs with 500 memristors per PM. The utility scores obtained when training each additional PM are shown in Fig. 4.11. By training the first PM, the utility score was found 0.430. This utility score corresponds to 77.4% correct predictions of non-sepsis labels and 72.1% correct predictions of sepsis labels. After training the second PM, the utility score of the two-PM system was worse 0.342. This happened because a system with two PMs favors the correct predictions of non-sepsis against the correct predictions of sepsis. Favoring correct non-sepsis predictions at the cost of correct sepsis predictions resulted in a decreased utility score: 91.4% correct predictions of non-sepsis labels against 45.1% correct predictions of sepsis labels. This is reasonable since a

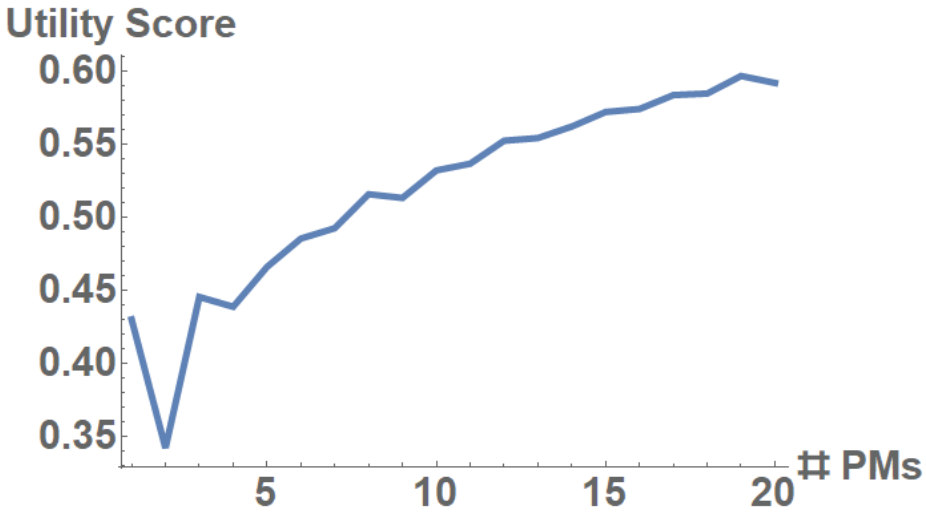


FIGURE 4.11: The obtained utility scores are shown on the vertical axis after training a system with PM. The number of PM per system is shown in the horizontal axes.

system with two PMs predicts sepsis only if both PMs predict sepsis. As more and more PMs were sequentially trained, the utility score had an increasing trend and a utility score 0.592 was obtained with a 20 PM system.

Chapter 5

Exploiting algorithms for efficient transient simulations

This chapter of this thesis summarizes the work done in papers III and IV. There has been a need to simulate accurately and efficiently several experimental elements. In this thesis, a generic electronic circuit simulator has been developed for simulating the transient behavior of electronic circuits with environment sensitive elements. The ultimate goal was to use the simulator as an optimization tool to identify optimal network designs (*e.g.* the drive signal and the network parameters). The simulator has been implemented as an integral part of an automatic genetic algorithm optimization procedure where many network designs are tested with some stochasticity until the one with a desired functionality is found. Since such a numerical optimization process involves an extremely large number of simulations, then, each simulation should be executed in a relatively short time, to make such an optimization approach feasible.

While many electronic component models have been imported to the simulator from the literature, *e.g.* the widely used models for resistor, memristor, capacitor, inductor, some components were modelled from scratch. In particular, a lot of effort has been put into developing efficient simulation algorithms for the constant phase element (CPE) and the organic electrochemical transistor (OECT). CPEs are models of electronic circuits that are used in equivalent electronic circuits of elements where ionic diffusion is involved. The OECTs are special purpose devices used for analyzing ionic solutions. There is a genuine lack of accurate and algorithmically efficient models for simulating OECT and CPE transient behaviors. Further, for the OECT element of interest, some models are available, but because of their special-purpose nature, they have severe limitations and could not be used directly.

In section 5.1, a new method for simulating the transient response of CPEs is given. In section 5.2, a generic theory of OECT transistors is given which can be used to develop a model for simulating the transient response of OECTs.

5.1 Transient simulation of electronic circuits with Constant Phase elements

The problem regarding the transient simulation of electrical circuits with CPEs is that a repeated numerical evaluation of a computationally expensive convolution integral is needed, shown in Eq. (5.1). This integral relates the instantaneous voltage drop $V_w(t)$ across the element, with the current that has passed through it $I_w(t')$ with $t' \leq t$. To avoid this problem, various methods have been suggested in the literature.

The standard method is to approximate the CPE element by an equivalent RLC circuit. [34, 35, 36, 37] These circuits are easier to simulate. For example, there are

commercial packages available that can be used to simulate them efficiently. However, these methods are only accurate in a short range of frequencies due to a finite (often a relatively small) number of resistors, capacitors or inductors. The accuracy in a wide range of frequencies requires the increase in the number of elements, and this increase implies a larger algorithmic complexity cost. It has been pointed out in [37] that the accurate approximation of RLC circuits in a wide range of frequencies is still an open problem.

Another set of methods focuses on expanding the convolution kernel as an infinite series of special functions. [38, 39] The advantage of these approaches is that if the series converges fast then only few terms in the series expansion need to be kept. However, in general it is hard to know how many terms should be kept.

A novel method has been developed for simulating the transient dynamics of CPEs that is both generic, remarkably efficient, and surprisingly accurate. For simplicity reasons this thesis focuses on a specific type of CPE, the Warburg element. The Warburg element is one type of CPE where the applied voltage difference V_w at time instance t, $V_w(t)$, and the current passing through it I_w at time instance t, $I_w(t)$ have a phase difference equal to 45 degrees. The method for simulating transient response of the Warburg element can be easily extended to CPEs.

The need for developing the new method emerges from the fact that the calculation of the voltage across the CPE, $V_w(t)$, requires the calculation of the following convolution integral [40]:

$$V_w(t) = \frac{A_w(\alpha_w)}{\Gamma(\alpha_w)} \int_0^t (t-u)^{\alpha_w - 1} I_w(u)\, du, \quad \alpha_w \in [0,1) \tag{5.1}$$

with Γ being the usual Gamma function, e.g. $\Gamma(1/2) = \sqrt{\pi}$, and $A_w(\alpha_w)$ is a device dependent constant.

Calculating numerically the convolution integral is reasonable for short time intervals. However, a repeated numerical evaluation of the convolution integral can be very expensive for long times. Problems arise regarding the memory usage and the computation time because the time instances of the current $I_w(t)$ have to be stored for a long time interval $[0, t]$. Additionally, the larger this time interval is, the more computationally expensive the calculation of this integral becomes. [40] For example, assuming a grid of n time points $\{t_0 = 0, t_1, \cdots, t_m, \cdots, t_n\}$, the computational cost of evaluating the convolution integral scales as

$$O(n^2) \sim \sum_{m=0}^{n} O(m) \tag{5.2}$$

where $O(m)$ is the algorithmic cost of evaluating the integral for a fixed time instance t_m. Paper III suggests a generic method for decreasing the algorithmic complexity by one order of magnitude.

5.1.1 Updating the convolution integral

For the Warburg element, $\alpha_w = 1/2$ and the voltage is calculated as the following convolution ('*' denotes the convolution operation):

$$V_w(t) = \frac{A_w}{\sqrt{\pi}} \frac{1}{\sqrt{t}} * I_w[t] = \frac{A_w}{\sqrt{\pi}}\, \Phi_w(t) \tag{5.3}$$

where $A_w \equiv A_w(\alpha_w = 1/2)$.

The computational cost of any standard quadrature algorithm for the approximation of the convolution integral $\Phi_w(t_n)$ in a discrete grid of n time-points $\{t_0, t_1, \cdots, t_m, \cdots, t_n\}$ scales with the size of the time grid n, where $t_0 = 0$ and the distance between two successive time-points is given as $\Delta t_i = t_i - t_{i-1}$:

$$\Phi_w(t_m) \approx \sum_{j=0}^{m} w_j^m \, I_{w,j} \tag{5.4}$$

where for further convenience the following notation is used: $I_{w,j} = I_w(t_j)$ and, the weight w_j^m depends on the time-points of the grid t_j and t_m.

While the computational cost of evaluating Eq. (5.4) at a fixed time instance t_m is $O(m)$, the problem is that a repeated evaluation for many time instances leads to a quadratic cost $O(n^2)$, and here it is assumed that the time grid of the simulation consists of n time points. To deal with this problem, in paper III, a method has been developed to calculate the convolution integral at time t_m, $\Phi_w(t_m)$, by using the already calculated $\Phi_w(t_m - dt)$. To do that, the convolution integral $\Phi_w(t_m)$ is split in two parts:

$$\Phi_w(t_m) = H(t_m, t_\lambda) + \Psi_0(t_m, t_\lambda) \tag{5.5}$$

where,

$$H(t_m, t_\lambda) = \int_{t_\lambda}^{t_m} \frac{I_w(u)\, du}{\sqrt{t-u}} \tag{5.6}$$

and,

$$\Psi_0(t_m, t_\lambda) = \int_{0}^{t_\lambda} \frac{I_w(u)\, du}{\sqrt{t-u}} \tag{5.7}$$

where $H(t_m, t_\lambda)$ is calculated with high precision and $\Psi_0(t_m, t_\lambda)$ is calculated by a simple update of the previously calculated $\Psi_0(t_{m-1}, t_{\lambda-1})$.

The key idea is shown in Fig. 5.1. The One part, $\Psi_0(t_m, t_\lambda)$, is used to approximate the convolution integral between 0 and t_λ and the other part, $H(t_m, t_\lambda)$, for the approximation between t_λ and t_m. At the next time step t_{m+1}, the convolution integral is written again a sum of the two parts:

$$\Phi_w(t_{m+1}) = H(t_{m+1}, t_{\lambda+1}) + \Psi_0(t_{m+1}, t_{\lambda+1}) \tag{5.8}$$

Notice here that the successive distances $t_m - t_\lambda$ and $t_{m+1} - t_{\lambda+1}$ are approximately equal. Therefore, the calculation of $H(t_m, t_\lambda)$ would require similar algorithmic complexity to the calculation of $H(t_{m+1}, t_{\lambda+1})$. However, the calculation of $\Psi_0(t_{m+1}, t_{\lambda+1})$ requires larger algorithmic complexity than the calculation of $\Psi_0(t_m, t_\lambda)$ because $t_{\lambda+1} > t_\lambda$.

Since the algorithmic complexity of calculating $H(t_m, t_\lambda)$ is similar at every time point t_m, we calculate this part with high precision. However, the algorithmic complexity of calculating $\Psi_0(t_m, t_\lambda)$ increases as t_m increases.

According to the developed method in paper III, Ψ_0 at the next time point, $\Psi_0(t_{m+1}, t_{\lambda+1})$, can be approximated by a simple update of Ψ_0 at the previous time point, $\Psi_0(t_m, t_\lambda)$. For this purpose, a dynamical system has been suggested with $N+1$ equations and a closure function. This dynamical system is shown in Eqs. (50) and (51) in paper III. The definition of the closure function is given in Eq. (49) in paper

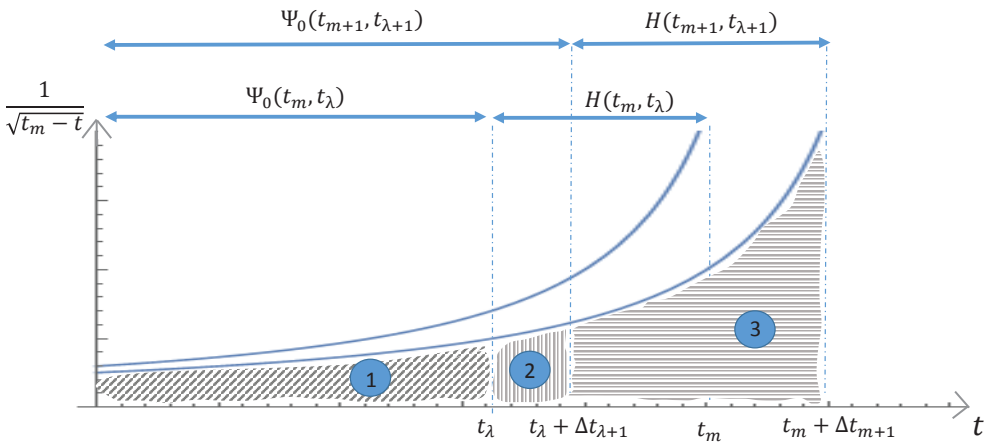


FIGURE 5.1: Figure taken from paper III. The concept of calculating the convolution integral at the time-point $t_m + \Delta t_{m+1}$ by using the previous convolution integral at the time-point t_m. The integral $\Phi(t_m)$ ($\Phi(t_m + \Delta t_{m+1})$) is calculated by convolution between the current and the curve reaching the time-point t_m ($t_m + \Delta t_{m+1}$). The integral $\Phi(t_m + \Delta t_{m+1})$ is approximated by three different calculations. Region 1 refers to the tail window calculation by using information from the previous convolution integral. Regions 2 and 3 refer to analytical calculation of the convolution integral by linear interpolating the current $I_w(t)$.

III:

$$r(t_{m+1}, t_\lambda) = \frac{\int_0^{t_\lambda} \frac{I_w(u)\ du}{(t_{m+1}-u)^{\frac{2\cdot N+3}{2}}}}{\int_0^{t_\lambda} \frac{I_w(u)\ du}{(t_{m+1}-u)^{\frac{2\cdot N+1}{2}}}} \tag{5.9}$$

One of the questions of this paper is how to calculate the closure function above. To calculate it analytically, it has been chosen that the current is constant $I_w(u) = I_o$.

Finally, an algorithm is suggested for performing transient simulations of electronic circuits with Warburg elements by using the Modified Nodal Analysis (MNA). The Warburg element is suggested to be used similarly to a Voltage source object with the MNA. The MNA is widely used in electronic circuit simulators for transient simulations and therefore the integration of the developed method with electronic circuit simulators is amenable.

5.1.2 Results

In paper III the designed algorithm was tested on a simple circuit driven by a voltage source. Different numerical simulations were performed for two different cases of the voltage source signal: DC and AC signals of different frequencies. The size of the dynamical system was set for all the simulations as $N = 1$.

By performing those numerical simulations, the effect of the distance $t_m - t_\lambda$ was investigated on the algorithmic complexity and the error. It was found that there is a trade-off between the error and the algorithmic complexity. By choosing the distance $t_m - t_\lambda$ one can regulate this trade-off. By increasing the distance $t_m - t_\lambda$, then, the error decreases at the cost of a larger execution time of the algorithm.

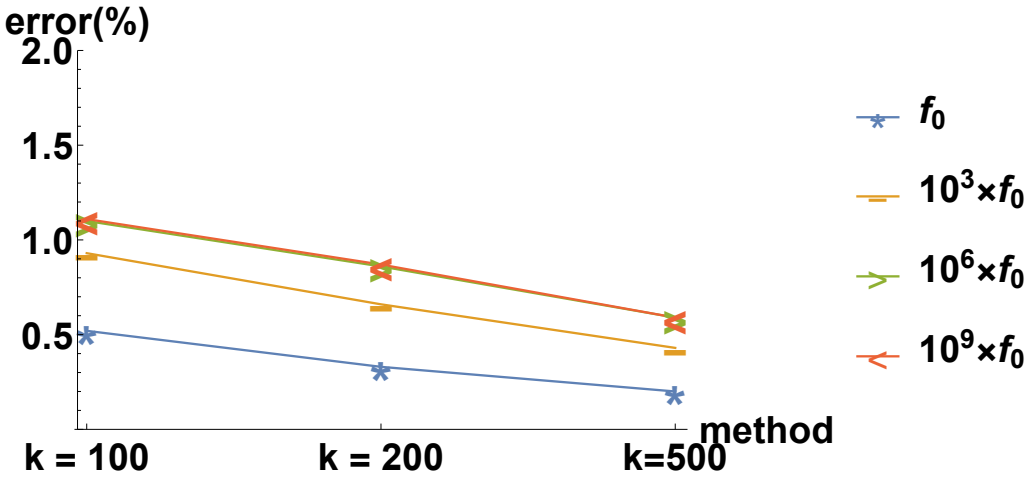


FIGURE 5.2: Figure taken from paper III. The error(%) when the circuit was simulated with an AC voltage source as a sinus signal with amplitude $0.001V$ and four different cases: with period $1/f_0 = 0.628s$, $1/(10^3\ f_0)$, $1/(10^6\ f_0)$ and $1/(10^9\ f_0)$ respectively. The time-step dt was such that 100 time-points were sampled per sinus cycle. The circuit was simulated for five cases: $k = 30$, $k = 100$, $k = 200$, $k = 500$ and $k = \infty$. The parameter k denotes the number of time-points which have been used to calculate $H(t_m, t_\lambda)$ with high precision. The error is calculated as the absolute difference between the ideal case $k = \infty$ and the every other case $k = 30, 100, 200, 500$ divided by the maximum value of the simulation for $k = \infty$. In the time domain, the error is oscillating from 0 to a maximum value. The maximum error is depicted on the vertical axes. The error(%) when $k = 30$ is not shown in this figure for resolution reasons. This error was found as 2.54%, 4.17%, 4.69% and 4.71% at the four frequencies respectively.

By comparing our method with a simple RC (resistor-capacitor) circuit, it was found that the execution time of our method is similar to a Three-RC circuit (with three capacitors and three resistors). This finding is interesting because one would expect a much larger execution time with our method since our method is heavily dependent on the approximation of $H(t_m, t_\lambda)$ being computed with high precision. However, the execution time with RC methods is relatively large due to the larger number of nodes in the equivalent circuits (6 nodes were used with the Three-RC circuit and 3 nodes with our method).

A key result is that our method is stable at a large range of frequencies ($1Hz - 1GHz$) as it is shown in Fig. 5.2. This is a great advantage of our method: The RLC circuits have equivalent impedance to constant phase elements in a small range of frequencies while our method is stable in very large range of frequencies. If one wanted to use an "RC circuit" method for achieving a low error in a larger range of frequencies, then, one should design an RC circuit with more components and voltage nodes but the algorithmic complexity of simulating such a circuit would be heavily increased.

5.2 Transient simulation of electronic circuits with Organic Electrochemical Transistors

Transient simulations of electrical circuits with OECTs can be a useful tool for designing efficient environment sensitive networks for biosensor applications. Such simulations can provide mechanistic understanding of the underlying physical concepts too. For example, they can be used to infer circuit parameters by fitting simulations to experimental data, etc.

In the literature, there have been successful theoretical models which unravel the underlying principles of OECTs. [41, 42, 43, 44] However, there are limitations regarding the usage of these models for building simulator primitives that are easily integrated in electric circuit simulators, *e.g.* such as SPICE.

For example, Faria et al. [44] have proposed a model for the drain current transient response. Their model is useful for predicting the transient drain current in a range of time when the gate voltage input is known in the whole range before the start of the simulation. However, when connecting OECTs in a network then the gate voltage cannot be known in the whole range. As an example, the gate voltage of one OECT might be dependent on the chemical concentrations around other OECTs, and therefore this gate voltage cannot be known before the simulation. Sideris *et al* [45] have also suggested a method for simulating the OECT transient. Their method is based on polynomial approximations of the drain current.

However, in literature a theoretical model of OECTs has not been found which is represented mathematically as a recurrent cell. In particular, there is a need for a model M_{oect} which should estimate the three currents at time t, $I_{oect}(t)$, (drain, source and gate current) provided the three currents at time $t - dt$, $I_{oect}(t - dt)$ and the three electrode voltages at time t, $V_{oect}(t)$, and $t - dt$, $V_{oect}(t - dt)$:

$$I_{oect}(t) = M_{oect}\left(I_{oect}(t - dt), V_{oect}(t), V_{oect}(t - dt)\right) \tag{5.10}$$

In paper IV, such a model has been suggested. This approach is more generic than the one by Sideris *et al*, since it is based on a genuine dynamic ordinary differential equation (ODE) paradigm, and allows for more flexible numerical integration techniques. For example, with a model M_{oect} it is possible to simulate networks of OECTs inter-connected to each others or even networks where OECTs are interconnected to other electronic elements.

5.2.1 Equations of motion

In paper IV, the equations of motion, which were previously developed by Bernards and Malliaras [41] have been generalised, and a system of partial differential equations has been obtained that describes how ionic degrees of freedom are coupled with the electrical degrees of freedom in the material.

The geometry of both the device and the electrolyte are shown in Fig. 5.3. The device is a semiconducting substrate with dimensions $\{L, w, y\}$ and is covered by ionic solution above. The device volume is divided in vertical slices with infinitesimal volumes, as shown in Fig. 5.3a. Every infinitesimal volume is described by using the equivalent circuit model, as shown in Fig. 5.3b. The gate voltage $V_g(t)$ is applied on the top of the electrolyte. The voltage on the boundary between the semiconductor and the electrolyte at the position x and time t is denoted as $V_{ch}(x, t)$.

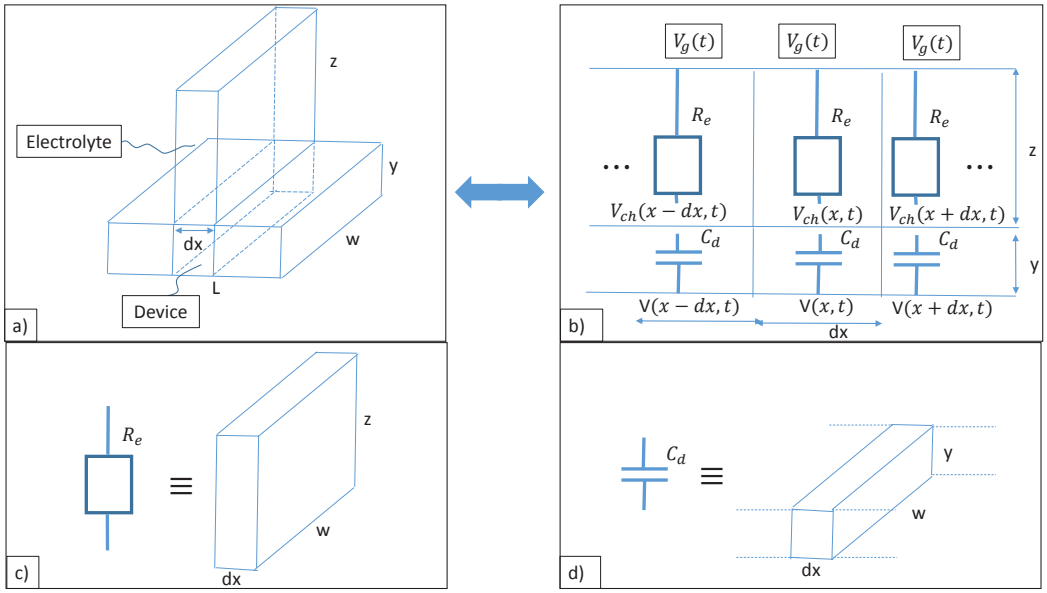


FIGURE 5.3: Figure taken from paper IV. a) The geometry of the electrolyte and the device is divided in infinitesimal slices with length dx. Each slice contains two parts. The volume $dx\ w\ y$ occupies the region in the OECT material. Above this volume, there is a volume of electrolyte $dx\ w\ z$. b) The equivalent circuits of the device and the electrolyte. The material acts as a volume capacitance where C_d is the capacitance of the device sub-volume. The electrolyte sub-volume has a resistance R_e. Applied voltages are: the gate voltage V_g, $V_{ch}(x,t)$ is the time-dependent voltage at the boundary between electrolyte and device at the position x, and $V(x,t)$ is the local voltage in the device sub-volume. c) The electrolyte is modeled by an equivalent resistance R_e. d) The device is modeled by an equivalent capacitor C_d.

$V(x,t)$ denotes the voltage inside the electrolyte. This simple model features a resistance R_e coupled in series to a capacitor C_d. The resistance describes the flow of ions through the slice of the electrolyte above the semiconductor (Fig. 5.3c).

After passing the electrolyte, the ions enter into the semiconductor material. It has been argued that a good model which describes this process is a volume capacitance. The capacitance of the piece of material with volume $v = ywdx$ is given by $C_d = c_d v$ where c_d is the volume capacitance of the device material (Fig. 5.3d).

In every small volume of the device there is an accumulated charge density $Q(x,t)$. In the equivalent circuit, this charge density is the charge of the capacitor C_d. By solving Kirchhoff laws in the equivalent circuits, the following dynamical equation is derived:

$$\frac{\partial Q(x,t)}{\partial t} = -\frac{Q(x,t)}{\tau} + \frac{1}{\tau}\, c_d\, y\, w\, [V_g(t) - V(x,t)] \tag{5.11}$$

with the time constant of the equivalent circuit given by $\tau = r_e c_d z y$.

Across the semiconductor material, the Ohm's law relates the current density $J(x,t)$ flowing through the semiconductor device and the voltage $V(x,t)$.

$$J(x,t) = -e\mu\rho(x,t)\frac{\partial V(x,t)}{\partial x} \tag{5.12}$$

where $\rho(x,t)$ is the local density of charge carriers, e is the charge of the carrier, and μ is their mobility. The free charge carrier density $\rho(x,t)$ is regulated by the concentration of ions $Q(x,t)$ that are absorbed in the material: an increase in $Q(x,t)$ leads to a decrease in $\rho(x,t)$. An approximate relationship between ρ and Q has been suggested in [41]:

$$\rho(x,t) = \rho_0 \left(1 - \frac{Q(x,t)}{Q_{max}}\right) \tag{5.13}$$

where ρ_0 and Q_{max} are device parameters.

Equations (5.12) and (5.11) have been solved and analyzed by making the assumption that the transient charge in the device $Q(x,t)$ can be approximated by the charge density at the steady state condition $Q_{st}(x,t)$ times a variable T:

$$Q(x,t) \approx T(t) Q_{st}[x, \xi(t)] \tag{5.14}$$

The variable T denotes how far the system is from the steady state condition. If $T = 1$, then the system is at the steady state. Otherwise, the system has less charge ($T < 1$) or more charge ($T > 1$) than in the steady state condition. The externally controlled electrode voltages: gate Voltage $V_g(t)$, drain voltage V_d and source Voltage V_s are collectively denoted as $\xi(t)$. These voltages determine the stationary state charge density profile. If ξ is altered, the charge density profile $Q_{st}(x)$ changes too, and to emphasize this we use $Q_{st}(x,\xi)$

After the straight forward but somewhat tedious algebra, which will not be reproduced here, the solution of the Eq. (5.12) by using the above assumption results in an analytical equation for the drain current I_D:

$$\begin{aligned} I_D(t) &= f_O[T(t), \xi(t)] = \\ &= G \left(1 - \frac{V_g - \frac{V_d}{2}}{V_p}\right) \frac{(V_d\, T)^2}{T\, V_d - \left[V_p\, (1-T)\right] Log\left(1 + \frac{T\, V_g}{V_p - T\, V_g}\right)} \end{aligned} \tag{5.15}$$

where G is the conductance of the semiconductor given as $G = e\mu\rho_o W \frac{Y}{L}$ and V_p is the pinch-off voltage of the semiconductor.

The analysis of the Eq. (5.11) with the above assumption considering a discrete time grid with time step Δt results in the following update rule for the parameter T:

$$T(t) = \frac{\tau}{\tau + \Delta t}\, \Lambda(t, \Delta t)\; T(t - \Delta t) \;+\; \frac{\Delta t}{\Delta t + \tau}\, \Xi(t) \tag{5.16}$$

where

$$\Lambda(t, \Delta t) \;\equiv\; \langle \Lambda(x, t, \Delta t) \rangle_x \tag{5.17}$$

$$\Xi(t) \;\equiv\; \langle \Xi(x, t) \rangle_x \tag{5.18}$$

with

$$\Lambda(x, t, \Delta t) = \frac{Q_{st}[x, \xi(t - \Delta t)]}{Q_{st}[x, \xi(t)]} = \frac{V_g(t - \Delta t) - V_{st}[x, \xi(t - \Delta t)]}{V_g(t) - V_{st}[x, \xi(t)]} \tag{5.19}$$

and

$$\Xi(x, t) = \frac{V_g(t) - V[x, T(t), \xi(t)]}{V_g(t) - V_{st}[x, \xi(t)]} \tag{5.20}$$

Herein, due to the fact that it is computationally expensive to calculate the integrals in Eqs. (5.17) and (5.18), $\Lambda(t, \Delta t)$ and $\Xi(t)$ are calculated by setting $x = \frac{L}{2}$ in Eqs. (5.19) and (5.20). It is not set $x = 0$ or $x = L$ because there are not transient dynamics at the bounds.

The parameter $\Xi(t)$ indicates how far the transient dynamics is from the steady state conditions. If $V[x, T(t), \xi(t)] = V_{st}[x, \xi(t)]$, then, $T = 1$ and $\Xi(t) = 1$, otherwise, $\Xi(t) \neq 1$ and the update rule has a tendency to move T towards 1.

The parameter $\Lambda(t, \Delta t)$ indicates if the steady state conditions have changed. If $\xi(t - \Delta t) = \xi(t)$, then, $\Lambda(t, \Delta t) = 1$, otherwise, $\Lambda(t, \Delta t) \neq 1$. This means that if $\Lambda(t, \Delta t) \neq 1$ then $\Lambda(t, \Delta t)\; T(t - dt) \neq T(t - dt)$ and the update rule is done from a different point of view.

However, up to here, the contribution of the current coming through the gate, gate current, has not been considered to contribute to the drain current. In previous works, it has been assumed that when the steady state conditions change, then a specific amount of the gate current is driven to the drain and the rest to the source electrode. This gate current is the reason for spikes observed experimentally in the drain current. [46, 44] Therefore, the total current through the drain electrode $I_{D,tot}$ would be calculated by adding a portion of the gate current $\Delta I_g(t)$:

$$I_{D,tot}(t) = I_D(t) + \alpha_1\, \Delta I_g(t) \tag{5.21}$$

where $0 < \alpha_1 < 1$, while $(1 - \alpha_1)\, \Delta I_g(t)$ flows into the source electrode. It has been assumed that the gate current is given by

$$\Delta I_g(t) = \frac{Q_{st,tot}(t) - Q_{st,tot}(t - dt)}{dt} \tag{5.22}$$

with the notation $Q_{st,tot}(t) \equiv Q_{st,tot}[\xi(t)]$, where

$$Q_{st,tot}[\xi(t)] \equiv \int_0^L dx Q_{st}[x, \xi(t)] \tag{5.23}$$

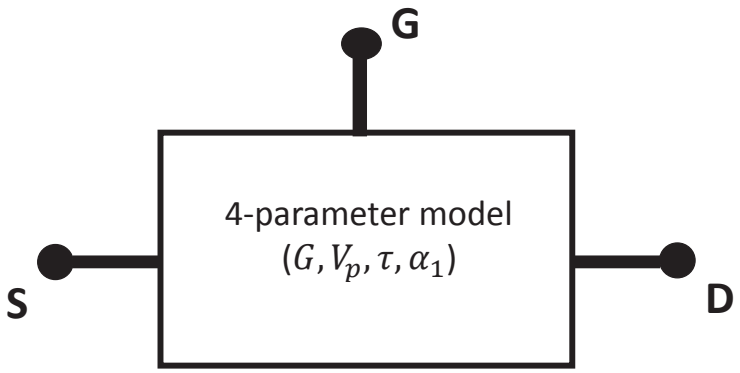


FIGURE 5.4: The OECT four-parameter model suggested in this thesis; S, G and D denote the electrode voltage nodes at the source, gate and drain respectively.

Finally, an algorithm is introduced for the transient simulation of OECT models connected to an electronic circuit by using the MNA [47]. The key primitive of the MNA paradigm is the idea of a stamp, as explained in paper III. The stamp of the OECT element is represented by three voltage dependent current sources: one current source at the drain node with the total drain current given by Eq. (5.21), one current source at the gate node with the current given as $-\Delta I_g(t)$ in Eq. (5.22) and one current source at the source node given as $-I_D + \Delta I_g(1-\alpha_1)$.

The four parameters of the model: The OECT model developed in this thesis is parameterized by the following quantities: the conductance G, the pinch-off voltage V_P, the time constant τ and the parameter α_1 as shown in Fig. 5.4.

5.3 Results

An example of fitting the model to experimental data is shown in Fig. 5.5. The experimental drain current was recorded by collaborators in the RECORD-IT project by using the methods given in [48, 49]. The experimental setup: $V_s = 0V$, $V_d = -0.1V$ and the gate voltage was pulsed with a square wave pulse of two levels $0V$ and $0.3V$ with 50% duty cycle. The four parameters of the model have been optimized so as the simulated total drain current $I_{D,tot}$ fits the experimental data.

As one can see in Fig. 5.5, the simulated total drain current agrees with the experimental one. The spike behaviour is correctly reproduced, both the onset and the recovery phases. Additionally, the simulated output relaxes towards the same steady state condition as the one in the experiment. However, there is a behavior that cannot be explained by the current model. In the experiment, the upward spikes are larger than the downward ones. The theoretical model predicts a fully symmetric behavior. For example, when the gate voltage increases from $0V$ to $0.3V$ and when it decreases from $0.3V$ to $0V$, then, the spike currents should have the same absolute value according to the analytical equations derived.

We also noticed in numerical simulations, that when the total current converges towards the steady state solution, then, the variable T converges towards 1. Additionally, when the total current decreased towards the steady state current, the variable T had larger values than one $T > 1$ and when the total current increased, then, it was that $T < 1$. This behavior of T shows that the dynamical system is stable.

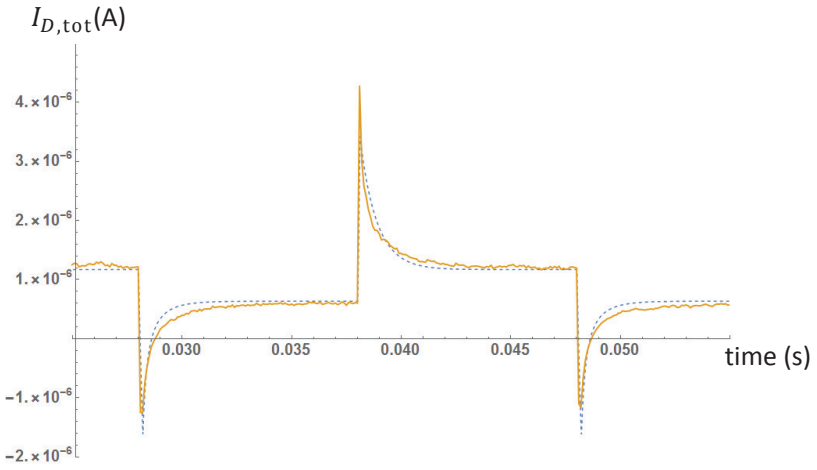


FIGURE 5.5: The simulated and the experimental drain current, with the theoretical parameters fitted to the experimental data. Dashed line: experimental data; The full line: the theoretical prediction.

In paper IV, a theoretical approach has been developed for simulating the transients of electrical circuits that contain OECT elements. The dynamical system is stable since the parameter T always converges to $T = 1$ when the external voltages are kept constant. Thus the OECT element always attains the steady state if given enough time, which is an important consistency check. The developed model is relatively easy to integrate into an arbitrary electric circuit simulator, especially the ones that use the MNA method.

The model describes nicely the experimental data, apart from predicting symmetric spike currents. It is possible that the asymmetric spike currents observed in the experiment, are a result of the low sampling frequency of the experimental data, and are in fact symmetric in reality. It would be interesting to test against experimental data collected with a larger sampling frequency. Further, regarding the relaxation to the state condition, we speculate that it might be that there are different time decay constants for the two different phases of increasing and decreasing towards the steady state conditions.

The model that has been developed can be used for reverse engineering of the OECT operation to understand underlying principles of the OECTs. Additionally, the fitting of the model to data can be used to investigate the sensitivity of the four OECT parameters (Fig. 5.4) to environmental conditions. For example, inferring the dependencies of the model parameters on chemical concentrations is a crucial task necessary for designing efficient sensors. These can be extracted by fitting the models to data for different ionic concentrations.

Chapter 6

Summary of appended papers

In paper I, the possibility of using environment sensitive memristor networks for advanced sensing applications has been investigated theoretically. The simplest possible memristor network has been considered which is a single environment sensitive memristor unit. The key challenge was to find an external drive signal so that the memristor's resistance is driven to two different regions when the memristor is exposed to two different environment signals. Two drive signals were found. The first drive signal was found by an intuitive analysis and the second drive signal was found by a genetic algorithm optimisation. For both drive signals, a synchronisation was found between the drive signals and the environment signals so that resistor's memristance was separable. In this paper, it was also shown that if this synchronisation is lost then memristance was not separable anymore. Finally, a trade-off between response time and phase space separation was shown. When the power of the drive signal increased, then, the response time increased at the cost of a reduced phase space separation.

In paper II, a separability index has been used as measure of the phase space separation. The separability index should be large given that the information embedded in the environment is highly correlated to the system state. This index has been defined in the context of this thesis without considering a specific readout layer. The separability index has been computed for a range of memristor networks with an increasing degree of complexity. The architectural features of memristor networks were identified which guarantee large phase space separation. The presence of feedback loops and increased degree of network heterogeneity were found to have the most impact on phase space separation.

In paper III, new methods have been developed and tested for performing transient simulations of electrical circuits that contain constant phase elements (CPEs). The problem is that the numerical evaluation of a convolution integral describing the response of a constant phase element is computationally very expensive with algorithmic complexity $O(n^2)$ for a grid with n time-points. The developed methods in paper III reduce the algorithmic complexity by one order of magnitude. Those objects can be used to integrate a CPE object in a simulator operating with Modified Nodal Analysis in a broad range of frequencies. This has not been achieved before. For example by using other methods (RLC equivalent circuits), one should consider a quite large circuit to operate with equivalent impedance to CPEs in the range of frequencies $1Hz - 1GHz$ which requires large execution times.

In paper IV, a dynamical model has been proposed for simulating the transient current responses of OECT devices. This model is advantageous since it can be simulated without any prior knowledge of the voltages at the electrodes. It can be used for transient simulations of electrical circuits consisting of many OECT devices coupled with other elements. It can be also used for reproducing the current behavior of OECT devices with different geometries: the device geometry of the model has been

fitted to experimental data of a device with different geometry. Additionally, this model reproduces the spike-and-recovery phase of the experimental OECT drain currents. However, it cannot reproduce some fine features of the experimental drain current behavior such as the experimental increasing spikes have larger magnitude than the experimental decreasing ones. Finally, the provided model is a generic template with separate modules and each module can be separately treated. As an example, a different model for the spike and recovery phase can be considered, for the ionic degrees of freedom or for the charge density distribution. All these can be easily integrated in future work.

In paper V, a new method has been introduced for comparing reservoirs with regards to their ability to separate inputs. Numerical experiments have been implemented on a series of memristor networks which have the same structure as some memristor networks considered in paper II. Both papers agree that input separation is favored by adding memristors in series and parallel. However, by adding continuously memristor elements in series and parallel is not expected to always increase the ability to separate inputs. This has been explained in details in the end of section 4.2 where it is concluded that other methods are needed to achieve maximum separation with the minimum number of memristor elements in a network.

In paper VI, a new method for training memristor models to predict sepsis at an early stage has been suggested. The results show that a single memristor with 79 trainable parameters as an input layer can achieve utility scores ≈ 0.3140 which correspond to approximately 80% true negatives and 58% true positives. This method is promising because it uses a few number of parameters but cannot be compared with other deep learning methods where thousand of parameters are trained. There is certainly underfitting to the training data. This thesis follows with a study on using memristor networks for overfitting the training data. The methods developed in paper VI are used to consider systems with heterogeneous memristors connected in parallel and only a trainable readout layer. The results of this thesis show that 500 memristors connected in parallel can achieve utility scores on the training data set comparable to other deep learning methods (≈ 0.43 utility score). This utility score was the maximum achieved on a separate test set in 2019 Physionet Computing in cardiology Challenge. This is a remarkable result because only 501 parameters have been trained. Another boosting algorithm has been also suggested in which one system can learn to correct the mistakes made by other systems. By using 20 systems consisting of 500 memristors each an even better overfitting has been obtained with utility score 0.59. Then, the next step would be to regulate overfitting and achieve a reasonable bias-variance trade-off. Since few parameters are trained in reservoir computing, it is expected that overfitting could be easily overcome, *e.g.* by using a larger training data set. The method developed in this thesis is generic and can be applied at other prediction problems or be extended even for regression problems.

In paper VII, a novel idea has been suggested for training chemical sensors to execute pattern recognition tasks of temporally extended variations of ionic concentrations. The idea has been demonstrated, both theoretically and experimentally, on a simple system consisting of only one chemical sensor and on the task of recognising simple patterns of zinc variations in a liquid. It has been shown that optimizing an auxiliary drive signal is a powerful resource for improving the sensor performance. A suitable drive signal was found so that the sensor's state becomes highly informative about patterns of ionic variations in the environment of the sensor. It has been shown that information about ionic variations can be correlated with shapes of state signals. This idea is combined with a data compression algorithm which converts signals into strings of characters where each character contains information about

the shape of the signal. The work featured a strong synergy between theoretical and experimental effort. The optimal drive signal was found through numerical simulations, where response of a theoretical model sensor has been studied under many conditions. This was the drive used in experiments. In the absence of a theoretical model, one could envision an experimental drive optimization procedure where a sensor is investigated under specific conditions *in vitro*. Once the drive has been found, one can potentially use it for *in vivo* applications. By the theoretical demonstration in this paper, it was shown that the proposed method works in theory, *i.e.* strings of characters were found which occurred with a high probability of one class of environment. The experimental demonstration points towards the right direction, but further work is needed to confirm its success.

In paper VIII, a novel classification method has been suggested that can be used to increase the intelligence of pattern recognition devices. Intelligence that normally resides in a reservoir can be moved to an external drive signal. This would allow engineering reservoirs with less complexity but without reducing the computing capacity of the device. Additionally, the intelligence of an existing system which is not meant to be modified could be increased. In this paper, it has been shown that electrocardiogram signals can be classified as either healthy or diseased by a single memristor element with an optimised drive signal. This system has been trained with few training examples (80 signals), has been tested with 1480 signals and still performed very well. One reason for this performance could be the few number of training parameters, 10 parameters were used to train the drive signal and 2 parameters to train a feedback whenever it was used. It has been also shown that training feedback mechanisms or input layers significantly improved the performance of the device. Those options would be useful for software implementations since a few number of parameters need to be trained but would cost in energy resources in hardware implementations.

In brief, the main findings of this thesis are as follows:

- An optimised drive signal can maximise the correlation between reservoir state and the environment one wishes to analyse.
- A measure of reservoir computing capacity can be defined without the consideration of a readout layer (sensing goal).
- Connecting heterogeneous elements in a network is necessary for increasing the network's computing capacity.
- One memristor can be used to perform classification tasks with the success rate that matches other state of the art approaches (*e.g.* deep learning methods).
- A new method has been suggested for simulating electronic circuits with constant phase elements. This algorithm is stable in a broad range of frequencies (1 Hz - 1 GHz) and costs linear algorithmic complexity with respect to the time of simulation.
- A generic theory has been proposed for modeling organic electrochemical transistors transient dynamics. An important practical outcome of this work is a model object that can be incorporated into any electronic circuit time-domain simulator.

Bibliography

[1] G. E. Moore. “Cramming more components onto integrated circuits (Reprinted from Electronics, pg 114-117, April 19, 1965)”. In: *Proceedings of the Ieee* 86.1 (1998), pp. 82–85.

[2] R. Cavin et al. “Emerging Research Architectures”. In: *Computer* 41.5 (2008), pp. 33–37.

[3] T. Sienko. *Molecular Computing*. MIT Press, 2003. ISBN: 9780262194877. URL: https://books.google.se/books?id=RleBQgAACAAJ.

[4] M. Hiratsuka, T. Aoki, and T. Higuchi. “Enzyme transistor circuits for reaction-diffusion computing”. In: *Ieee Transactions on Circuits and Systems I-Fundamental Theory and Applications* 46.2 (1999), pp. 294–303.

[5] A. Hjelmfelt, E. D. Weinberger, and J. Ross. “Chemical implementation of finite-state machines”. In: *Proceedings of the National Academy of Sciences of the United States of America* 89.1 (1992), pp. 383–387.

[6] Christopher Bennett et al. “On the inverse pattern recognition problem in the context of the time-series data processing with memristor networks”. In: *Advances in Unconventional Computation*. Ed. by Andrew Adamatzky. Vol. Vol 2. Prototypes and algorithms. Springer, 2016.

[7] Erik Bergh and Zoran Konkoli. “On improving the expressive power of chemical computation”. In: *Advances in Unconventional Computation*. Ed. by Andrew Adamatzky. Vol. Vol 2. Prototypes and algorithms. Springer, 2016.

[8] Zoran Konkoli. “On reservoir computing: from mathematical foundations to unconventional applications”. In: *Advances in Unconventional Computation*. Ed. by Andrew Adamatzky. Vol. Vol 1. Theory. Springer, 2016.

[9] Herbert Jaeger. *The "echo state" approach to analysing and training recurrent neural networks*. Report GDM Report 148 (contains errors). German national research center for information technology, 2001.

[10] H. Jaeger and H. Haas. “Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication”. In: *Science* 304.5667 (2004), pp. 78–80.

[11] W. Maass, T. Natschlager, and H. Markram. “Real-time computing without stable states: A new framework for neural computation based on perturbations”. In: *Neural Computation* 14.11 (2002), pp. 2531–2560.

[12] H. Markram T. Natschläger W. Maass. “The "Liquid Computer": A Novel Strategy for Real-Time Computing on Time Series (Special Issue on Foundations of Information Processing)”. In: *TELEMATIK* 8.1 (2002), pp. 39–43.

[13] M. Lukosevicius, H. Jaeger, and B. Schrauwen. “Reservoir Computing Trends”. In: *KI - Könstliche Intelligenz* 26.4 (2012), pp. 365–371.

[14] Chao Du et al. “Reservoir computing using dynamic memristors for temporal information processing”. In: *Nature Communications* 8.1 (2017), p. 2204. ISSN: 2041-1723. DOI: 10.1038/s41467-017-02337-y.

[15] Hamedani Kian et al. "The Novel Applications of Deep Reservoir Computing in Cyber-Security and Wireless Communication". In: *Intelligent System and Computing*. 2019. DOI: 10.5772/intechopen.89328. URL: https://www.intechopen.com/online-first/the-novel-applications-of-deep-reservoir-computing-in-cyber-security-and-wireless-communication.

[16] Alice Zheng and Amanda Casari. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. O'Reilly Media, Inc., 2018, p. 218. ISBN: 1491953241, 9781491953242.

[17] Zoran Konkoli. *The state weaving environment-echo tracker (SWEET/RECORD-IT) sensing setup and algorithm (patent pending)*. Patent. 2016.

[18] Z. Konkoli. "The sweet algorithm: generic theory of using reservoir computing for sensing applications". In: *International Journal of Parallel, Emergent and Distributed Systems* (2016).

[19] G. Monte. "Sensor signal preprocessing techniques for analysis and prediction". In: *2008 34th Annual Conference of IEEE Industrial Electronics*, pp. 1788–1793. ISBN: 1553-572X. DOI: 10.1109/IECON.2008.4758225.

[20] Gouhei Tanaka et al. "Recent advances in physical reservoir computing: A review". In: *Neural Networks* 115 (2019), pp. 100 –123. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2019.03.005. URL: http://www.sciencedirect.com/science/article/pii/S0893608019300784.

[21] Kiran Kumar Tadi et al. "Oxytocin-Monolayer-Based Impedimetric Biosensor for Zinc and Copper Ions". In: *ACS Omega* 2.12 (2017), pp. 8770–8778. ISSN: 2470-1343. DOI: 10.1021/acsomega.7b01404. URL: https://doi.org/10.1021/acsomega.7b01404.

[22] Yong Yu et al. "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures". In: *Neural Computation* 31.7 (2019), pp. 1235–1270. DOI: 10.1162/neco_a_01199%M31113301. URL: https://www.mitpressjournals.org/doi/abs/10.1162/neco_a_01199.

[23] Gouhei Tanaka et al. "Recent advances in physical reservoir computing: A review". In: *Neural Networks* 115 (2019), pp. 100–123. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2019.03.005. URL: http://www.sciencedirect.com/science/article/pii/S0893608019300784.

[24] Zoran Konkoli. "On reservoir computing: from mathematical foundations to unconventional applications". In: *Advances in Unconventional Computing*. Ed. by Andrew Adamatzky. Vol. 1. Theory. Springer, 2016.

[25] Massimiliano Di Ventra et al. "Circuit elements with memory: memristors, memcapacitors, and meminductors". In: *Proceedings of the IEEE* 97.10 (2009), pp. 1717 –1724.

[26] S. Smaili and Y. Massoud. "Analytic modeling of memristor variability for robust memristor systems designs". In: *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2014, pp. 794–797. DOI: 10.1109/ISCAS.2014.6865255.

[27] R. Naous, M. Al-Shedivat, and K. N. Salama. "Stochasticity Modeling in Memristors". In: *IEEE Transactions on Nanotechnology* 15.1 (2016), pp. 15–28. ISSN: 1536-125X. DOI: 10.1109/TNANO.2015.2493960.

[28] Saul B. Needleman and Christian D. Wunsch. "A general method applicable to the search for similarities in the amino acid sequence of two proteins". In: *Journal of Molecular Biology* 48.3 (1970), pp. 443–453. ISSN: 0022-2836. DOI: https://doi.org/10.1016/0022-2836(70)90057-4. URL: http://www.sciencedirect.com/science/article/pii/0022283670900574.

[29] Alar Ainla et al. "A multifunctional pipette". In: *Lab on a Chip* 12.7 (2012), pp. 1255–1261. ISSN: 1473-0197. DOI: 10.1039/C2LC20906C. URL: http://dx.doi.org/10.1039/C2LC20906C.

[30] Sanjukta Krishnagopal, Yiannis Aloimonos, and Michelle Girvan. "Similarity Learning and Generalization with Limited Data: A Reservoir Computing Approach". In: *Complexity* 2018 (2018), p. 15. DOI: 10.1155/2018/6953836. URL: https://doi.org/10.1155/2018/6953836.

[31] Priyadarshini Panda and Narayan Srinivasa. "Learning to Recognize Actions From Limited Training Examples Using a Recurrent Spiking Neural Model". In: *Frontiers in neuroscience* 12 (2018), pp. 126–126. ISSN: 1662-4548 1662-453X. DOI: 10.3389/fnins.2018.00126. URL: https://www.ncbi.nlm.nih.gov/pubmed/29551962https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5840233/.

[32] Ma Reyna et al. "Early prediction of sepsis from clinical data: the PhysioNet / Computing in Cardiology Challenge 2019". In: *Critical Care Medicine* 48 (2020), pp. 210–217. DOI: 10.1097/CCM.0000000000004145.

[33] Raschka Sebastian and Mirjalili Vahid. "Predicting Continuous Target Variables with Regression Analysis". In: *Python Machine Learning*. Packt Publishing Ltd., 2019.

[34] N. A. Sekushin. "Equivalent Circuit of Warburg Impedance". In: *Russian Journal of Electrochemistry* 45.7 (2009).

[35] Y. Tsividis and J. Milios. "A detailed look at electrical equivalents of uniform electrochemical diffusion using nonuniform resistance-capacitance ladders". In: *Journal of Electroanalytical Chemistry* 707 (2013).

[36] J. Valsa J. Vlach. "RC models of a constant phase element". In: *INTERNATIONAL JOURNAL OF CIRCUIT THEORY AND APPLICATIONS* 41.1 (2013), pp. 59–67. ISSN: 0098-9886.

[37] O. Enacheanu et al. "Identification of fractional order models for electrical networks". In: *IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics*. 2006, pp. 5392–5396. DOI: 10.1109/IECON.2006.348151.

[38] Agnieszka Jakubowska and Janusz Walczak. "Analysis of the Transient State in a Series Circuit of the Class $RL_\beta C_\alpha$". In: *Circuits, Systems, and Signal Processing* 35.6 (2016), pp. 1831–1853. ISSN: 1531-5878. DOI: 10.1007/s00034-016-0270-2. URL: https://doi.org/10.1007/s00034-016-0270-2.

[39] Francisco Gómez, Juan Rosales, and Manuel Guía. "RLC electrical circuit of non-integer order". In: *Central European Journal of Physics* 11.10 (2013), pp. 1361–1365. ISSN: 1644-3608. DOI: 10.2478/s11534-013-0265-6. URL: https://doi.org/10.2478/s11534-013-0265-6.

[40] O. Kanoun. *Lecture Notes on Impedance Spectroscopy: Measurement, Modeling and Applications*. CRC Press, 2012. ISBN: 9780203610756. URL: https://books.google.se/books?id=31-dDQAAQBAJ.

[41] D./A. Bernards and G./G. Malliaras. "Steady-State and Transient Behavior of Organic Electrochemical Transistors". In: *Advanced Functional Materials* 17.17 (2007), pp. 3538–3544. ISSN: 1616-3028. DOI: 10.1002/adfm.200601239. URL: http://dx.doi.org/10.1002/adfm.200601239.

[42] Jonathan Rivnay et al. "High-performance transistors for bioelectronics through tuning of channel thickness". In: *Science Advances* 1.4 (2015). DOI: 10.1126/sciadv.1400251. eprint: http://advances.sciencemag.org/content/1/4/e1400251.full.pdf. URL: http://advances.sciencemag.org/content/1/4/e1400251.

[43] Jacob T. Friedlein et al. "Optical Measurements Revealing Nonuniform Hole Mobility in Organic Electrochemical Transistors". In: *Advanced Electronic Materials* 1.11 (2015). 1500189, 1500189–n/a. ISSN: 2199-160X. DOI: 10.1002/aelm.201500189. URL: http://dx.doi.org/10.1002/aelm.201500189.

[44] Gregório C. Faria, Duc T. Duong, and Alberto Salleo. "On the transient response of organic electrochemical transistors". In: *Organic Electronics* 45 (2017), pp. 215–221. ISSN: 1566-1199. DOI: https://doi.org/10.1016/j.orgel.2017.03.021. URL: http://www.sciencedirect.com/science/article/pii/S1566119917301313.

[45] P. Sideris, S. Siskos, and G. Malliaras. "Verilog-A modeling of Organic Electrochemical Transistors". In: *2017 6th International Conference on Modern Circuits and Systems Technologies (MOCAST)*. 2017, pp. 1–4. DOI: 10.1109/MOCAST.2017.7937645.

[46] Jacob T. Friedlein et al. "Microsecond Response in Organic Electrochemical Transistors: Exceeding the Ionic Speed Limit". In: *Advanced Materials* 28.38 (2016), pp. 8398–8404. ISSN: 1521-4095. DOI: 10.1002/adma.201602684. URL: http://dx.doi.org/10.1002/adma.201602684.

[47] Chung-Wen HO, ALBERT E. RUEHLI, and PIERCE A. BRENNAN. "The Modified Nodal Approach to Network Analysis". In: *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS* CAS-22 (1975). DOI: 10.1109/TCS.1975.1084079.

[48] Sébastien Pecqueur et al. "Concentric-Electrode Organic Electrochemical Transistors: Case Study for Selective Hydrazine Sensing". In: *Sensors* 17.3 (2017), p. 570. ISSN: 1424-8220. URL: http://www.mdpi.com/1424-8220/17/3/570.

[49] Sébastien Pecqueur et al. "Cation discrimination in organic electrochemical transistors by dual frequency sensing". In: *Organic Electronics* 57 (2018), pp. 232–238. ISSN: 1566-1199. DOI: https://doi.org/10.1016/j.orgel.2018.03.020. URL: http://www.sciencedirect.com/science/article/pii/S1566119918301265.